

# **Machine Learning**

## **INTERNSHIP REPORT**

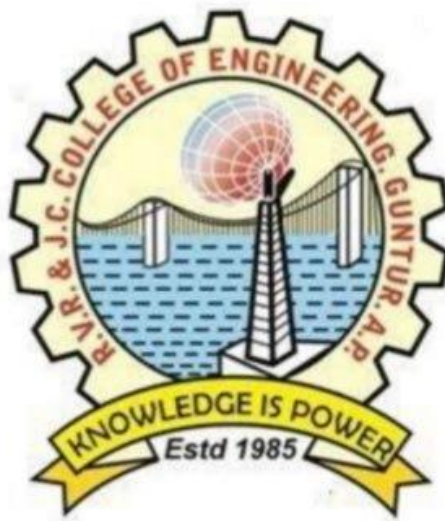
**Submitted in partial fulfilment of requirements to**

**Internship (CS-451)**

**IV /IV B. Tech CSE(VII Semester)**

**Submitted By**

**PURAM ANJALIDEVI(Y21CS136)**



**November 2024**

**R.V.R & J.C. COLLEGE OF ENGINEERING**

**(Autonomous)**

**Approved by AICTE :: Affiliated to Acharya Nagarjuna University**

**Chowdavaram, GUNTUR – 522019, Andhra Pradesh, India**

# Internship Certificate



**CERTIFICATE OF INTERNSHIP**

This is to Certify that Mr./Ms

**Anjalidevi Puram**

Enrolled in the Computer Science and Engineering - Y21CS136

From College R.V.R. & J.C.College of Engineering

of university Acharya Nagarjuna University

has Successfully Completed short-term Internship programme titled

**Machine Learning**

under SkillDzire for 2 Months.Organized By **SkillDzire** in collaboration  
with **Andhra Pradesh State Council of Higher Education.**

Certificate ID:  
**SDST-07099**

Issued On:  
**1-Jul-2024**



Approved By AICTE



Authorized Signature

## ACKNOWLEDGEMENT

The successful completion of any task must contains these three elements suggestions ,guidance and good environment . The combination of these three factors acts as backbone to our “**MACHINE LEARNING**”.

I'm express my sincere thanks to **Dr. M. Sreelatha** ,Head of the Department of Computer Science and Engineering, **Dr. K. Srinivas**, Principle of **R.V.R & J.C College of Engineering**, Guntur for their encouragement and support to carry out this project successfully.

I'm very Greatful to **SKILLDZIRE** ,Vijayawada for providing this supportive environment, I'm really thankful to my mentor **Mr. M. Srikanth** Sir for his guidance and support throughout the program.

Finally , I submit my reserves thanks to lab staff in Department of Computer Science and Engineering for their during the preparation.

**Puram Anjalidevi(Y21CS136)**

## **ABSTRACT**

Machine learning (ML) is a subset of artificial intelligence focused on enabling computers to learn from data and make decisions without explicit programming. It involves algorithms that identify patterns and relationships within datasets. ML is categorized into supervised, unsupervised, and reinforcement learning, each serving different applications. Supervised learning uses labeled data to predict outcomes, while unsupervised learning discovers hidden structures in unlabeled data. Reinforcement learning involves agents making decisions through trial and error to maximize rewards. The technology powers diverse applications, from image recognition and natural language processing to predictive analytics and autonomous systems. As data availability increases, ML's impact on various industries continues to grow, transforming how we analyze and interact with information. Challenges include ensuring data quality, addressing bias, and interpreting model decisions.

The Machine Learning field, which can be briefly defined as enabling computers make successful predictions using past experiences, has exhibited an impressive development recently with the help of the rapid increase in the storage capacity and processing power of computers. Together with many other disciplines, machine learning methods have been widely employed in bioinformatics. The difficulties and cost of biological analyses have led to the development of sophisticated machine learning approaches for this application area.

Contents	Page No
Title	i
Certificate	ii
Acknowledge	iii
Abstract	iv
Contents	v
List of Figures	vi
List of Tables	vi
<b>1 INTRODUCTION</b>	
1.1Background	1
1.2 Important terms in Machine Learning	1
1.3 Problem statement	2
<b>2 SYSTEM ANALYSIS</b>	
2.1 Requirements Specification	3
2.1.1 Functional Requirements	3
2.1.2 Non-Functional Requirements	4
<b>3 SYSTEM DESIGN</b>	
3.1 Architecture diagram	5
3.2 Types of MachineLaerning	9
3.2 Machine Learning Algorithms	14
<b>4 DATASET DESCRIPTION</b>	18
<b>5 IMPLEMENTATION AND TESTING</b>	21
<b>6 RESULT AND FINDING</b>	30
<b>7 CONCLUSION</b>	35
<b>8 REFERENCES</b>	36

## **List of Figures**

<b>S.No</b>	<b>FigureNumber</b>	<b>Figure Name</b>
1	3.1	ML Architecture
2	3.2	Machine Learning Types
3	3.3.1	Logistic Regression graph
4	3.3.2	Decission Tree
5	3.3.3	K-means Classifier
6	5.1	Covid Boolean graph
7	5.2	Percentage of Covid Positive

## **List of Tables**

<b>S.No</b>	<b>Table Number</b>	<b>Table Name</b>
1	4.1	Covid Dataset 1
2	4.2	Covid Dataset 2

# 1 INTRODUCTION

## 1.1 Background

Definition: A computer program is said to learn from experience  $E$  with respect to some class of tasks  $T$  and performance measure  $P$ , if its performance at tasks in  $T$ , as measured by  $P$ , improves with experience  $E$ . Machine learning (ML) has its roots in several disciplines, including statistics, computer science, and cognitive science. The idea of machines that can learn dates back to the 1950s, with pioneers like Alan Turing proposing algorithms that could improve from experience. Early work included the development of the Perceptron, one of the first neural networks. In the 1980s, interest in ML revived with the introduction of backpropagation, which allowed for training multi-layer neural networks. However, limitations in computational power and data availability constrained progress during this time. The advent of the internet in the 2000s led to an explosion of data, coupled with enhanced computing capabilities, which accelerated advancements in ML algorithms. The rise of deep learning in the 2010s transformed the field, allowing for significant improvements in tasks such as image and speech recognition. Frameworks like TensorFlow and PyTorch emerged, democratizing access to these powerful tools. Today, ML applications span diverse industries, from healthcare to finance, revolutionizing decision-making and operational efficiencies.

## 1.2 Important terms in Machine Learning

1. **Algorithm:** A set of rules or instructions for solving a problem or performing a task.
2. **Dataset:** A collection of data used for training and testing machine learning models.
3. **Training:** The process of teaching a model using a dataset to improve its accuracy.
4. **Test Set:** A subset of data used to evaluate the performance of a trained model.
5. **Feature:** An individual measurable property or characteristic of the data used in modeling.
6. **Overfitting:** A modeling error that occurs when a model learns noise from the training data, resulting in poor generalization to new data.
7. **Underfitting:** A scenario where a model is too simple to capture the underlying patterns in the data.
8. **Cross-Validation:** A technique for assessing how a model generalizes by splitting the dataset into multiple subsets for training and validation.
9. **Neural Networks:** A class of algorithms inspired by the human brain, commonly used in deep learning for complex tasks.
10. **Gradient Descent:** An optimization algorithm used to minimize the loss function by iteratively adjusting model parameters.
11. **Hyperparameters:** Settings or configurations that govern the training process but are not learned from the data.
12. **Classification:** A type of supervised learning where the model predicts categorical labels.
13. **Regression:** A type of supervised learning where the model predicts continuous values.

**14. Clustering:** An unsupervised learning technique that groups similar data points together without labeled outputs.

### **1.3 Problem Statement**

As we all know, how covid has rattled each and every one of us throughout the world and millions have died and still suffering due to losses occurred due to it. So, this project explains us and shows some insights about the symptoms of the covid persons, which helps us to identify whether a person is having covid/not by using machine learning algorithms. Using those as inputs we can develop an intelligent model which can help us to predict whether a person is affected with covid or not.



## 2 SYSTEM ANALYSIS

### 2.1.1 Functional requirements

Functional requirements specify what the system should do. In the context of machine learning, these could include:

- 1 Data Input and Output: The system should accept data in specified formats (e.g., CSV, JSON). The output should include predictions, classifications, or recommendations.
- 2 Model Training: The system should allow for the training of various ML models (e.g., regression, classification, clustering). It should support multiple training algorithms (e.g., decision trees, neural networks).
- 3 Evaluation Metrics: The system should provide methods for evaluating model performance (e.g., accuracy, precision, recall, F1 score).
- 4 Data Preprocessing: The system should include functionalities for data cleaning, normalization, and feature extraction.
- 5 Model Deployment: The system should allow for the deployment of trained models into production environments.
- 6 User Interaction: The system should include an interface for users to interact with the model (e.g., input data, retrieve predictions).

### 2.1.2 System Requirements

#### Software Requirements:

- Operating System: windows XP, WINDOWS 7, 8.1,10,11
- Web Browser: GOOGLE CHROME for Dataset
- Tools: Jupyter Notebook

**Hardware Requirements:** Personal computer with keyboard and mouse maintained with uninterrupted power supply.

- Processor: Intel® core™ i5
- Installed Memory (RAM): 8.00 GB
- Hard Disc: 250GB SSD

### 2.1.3 Non-Functional Requirements

Non-functional requirements define the quality attributes or constraints of the system. For machine learning, these might include:

1. Performance: The system should provide predictions within a specified time frame (e.g., less than 2 seconds for real-time applications).

2. Scalability: The system should be able to handle increasing amounts of data and users without significant performance degradation.
3. Reliability: The system should perform consistently under expected conditions and recover gracefully from failures.
4. Maintainability: The system should be easy to update with new data, features, or models without extensive downtime.
5. Security: The system should protect against unauthorized access to data and models, ensuring data privacy and compliance with regulations.
6. Usability: The user interface should be intuitive and easy to navigate for users with varying levels of expertise.
7. Interpretability: The system should provide insights into how decisions are made by the models, especially in critical applications (e.g., healthcare, finance).

## 3 SYSTEM DESIGN

### 3.1 Architecture

The diagram above focuses on a client-server architecture of a “supervised learning” system (e.g. classification and regression), where predictions are requested by a client and made on a server. (Side note: it might be preferable in certain systems to have client-side predictions; others might even motivate client-side model training, but the tools to make this efficient in an industrial ML application don’t exist yet.)

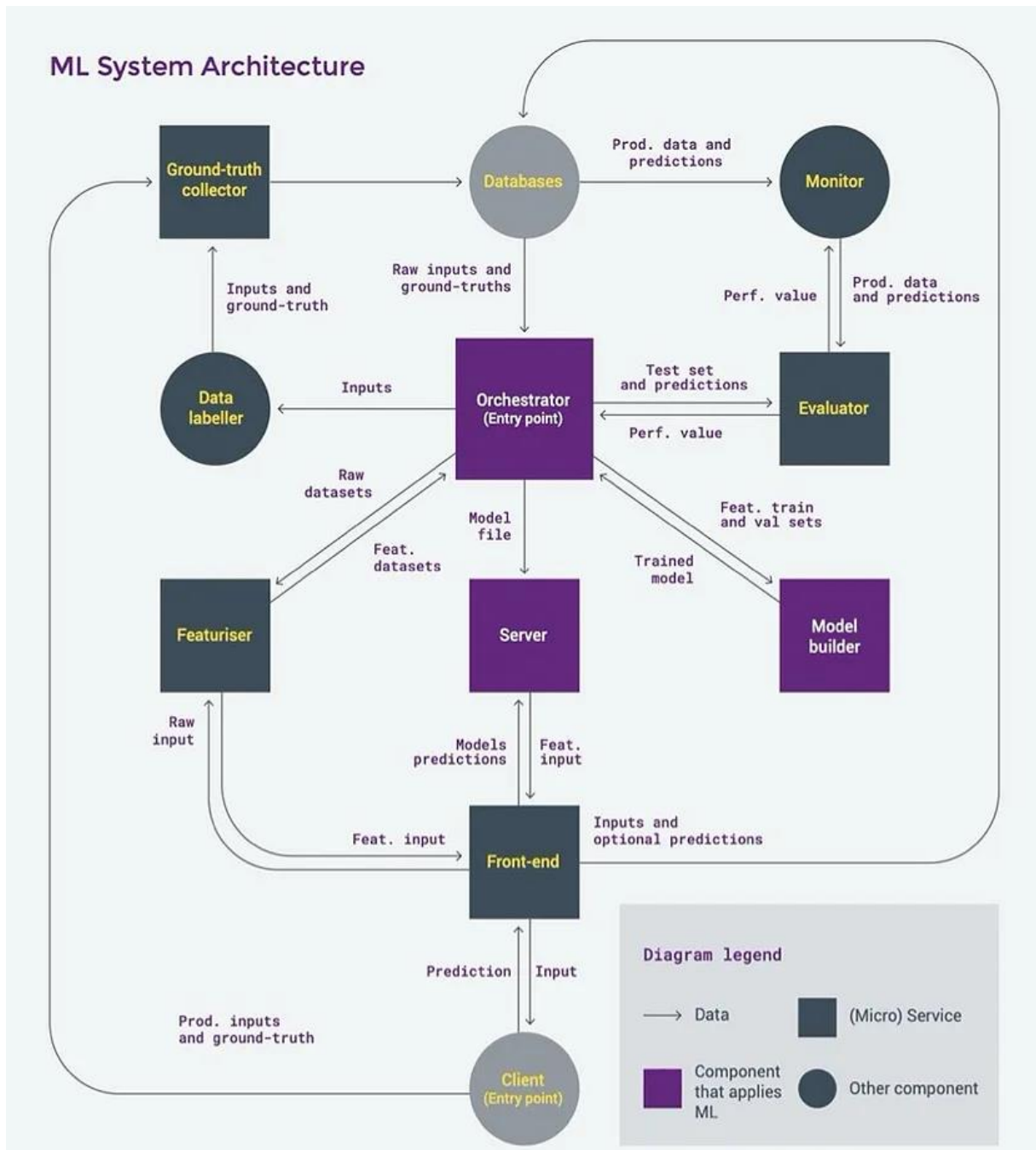


Fig 3.1 Architecture

## Data Ingestion

Data ingestion is a foundational process in machine learning (ML), referring to the acquisition and processing of data from various sources for use in ML models. It is the first step of the data pipeline and directly influences the quality and efficacy of the final models. Without a robust data ingestion pipeline, the downstream processes of data processing, transformation, and modeling can become ineffective. Data ingestion typically includes several sub-processes such as **data collection**, where raw data is gathered from multiple sources like databases, APIs, or sensors. The variety of sources—structured, semi-structured, or unstructured—can make ingestion challenging. Thus, it's important to employ appropriate methods to extract and load data that is both accurate and relevant to the problem being solved.

## Data Cleansing

Once data is collected, it often requires **data cleansing**. This step involves identifying and rectifying errors, inconsistencies, and missing values in the dataset. Raw data is rarely in the ideal form for direct analysis, and ensuring its quality is crucial for building reliable models. Data cleansing may involve removing duplicates, handling outliers, imputing missing values, and standardizing formats across different datasets. A clean dataset improves the performance of machine learning algorithms, as noise or irrelevant data can distort model training. The quality of the data directly impacts the accuracy of predictions, making cleansing an indispensable part of the ingestion process.

## Data Transformation

After data cleansing, the next step is **data transformation**. This refers to converting the cleaned data into a format that is suitable for analysis and modeling. Data transformation can include operations such as normalization, scaling, encoding categorical variables, and creating new features based on domain knowledge. The goal is to convert raw data into a structured form that helps the ML algorithms extract meaningful patterns. Proper data transformation is essential to enhance the learning process and ensure that models perform optimally. Additionally, transformed data needs to be compatible with various ML models, which may require different formats or data types.

## Data Integration

In many real-world scenarios, data comes from multiple, disparate sources. **Data integration** involves combining data from various systems, databases, and file formats into a unified dataset. This process can involve techniques such as data fusion, where data from different sources is merged into a single dataset while maintaining its consistency and integrity. Integrating data efficiently is key to providing a complete picture for model training. Whether it's customer data from CRM systems, transactional data from e-commerce platforms, or sensor data from IoT devices, integrating these data sources into a single coherent dataset is a critical task before training ML models.

## Data Sampling and Data Splitting

Once the data is prepared, **data sampling** and **data splitting** are essential processes to ensure the model can be trained, validated, and tested appropriately. Data sampling involves selecting a subset of the data for model training, which is particularly important in large datasets. This helps in reducing computational costs while maintaining model accuracy. On the other hand, **data splitting** involves partitioning the dataset into distinct sets for training, validation, and testing. The training set is used

to train the model, the validation set helps tune hyperparameters, and the test set is used to evaluate model performance. Proper splitting ensures that models are not overfitting and generalize well to unseen data.

## Real-time Data Ingestion

For certain applications, such as fraud detection or real-time recommendation systems, **real-time data ingestion** becomes essential. Unlike batch ingestion, which processes data in large chunks at scheduled intervals, real-time ingestion ensures that new data is continuously fed into the system as it becomes available. This is crucial for scenarios where the model needs to update its predictions frequently based on the latest data. Implementing real-time data pipelines requires sophisticated streaming technologies such as Apache Kafka, Apache Flink, or cloud-based services to handle high-velocity data without introducing delays in processing.

## Ingestion of Change Data Capture (CDC)

**Change Data Capture (CDC)** is a technique used to track and capture changes made to a dataset, such as new records, updates, or deletions. When applied to data ingestion, CDC ensures that only modified or new data is ingested, rather than reprocessing the entire dataset. This can significantly improve performance and reduce the time required for data ingestion, especially in systems with massive datasets that change frequently. CDC is particularly useful in scenarios where the dataset is dynamic, and keeping the ML model up-to-date with the latest changes is critical to maintaining its predictive accuracy.

## Data Storage

Once the data has been ingested, stored, and processed, the next challenge is **data storage**. The storage system must be selected based on the specific requirements of the ML project, including data volume, access speed, and processing complexity. Traditional relational databases might be adequate for smaller datasets, but for large-scale ML projects, distributed storage systems like Hadoop HDFS or cloud-based object storage such as Amazon S3 or Google Cloud Storage are often used. Choosing the right storage solution ensures that the data is easily accessible, can be efficiently queried, and integrates seamlessly with the ML pipeline. Moreover, as the models evolve, multiple versions of datasets and model artifacts need to be stored and managed properly.

## Data Version Control

A key aspect of modern machine learning is **data version control (DVC)**. Just as code versioning is essential for software development, versioning the data used for training models ensures that experiments are reproducible and can be tracked over time. DVC allows teams to maintain a historical record of datasets, transformations, and model parameters. This is crucial for collaboration, experimentation, and ensuring that any changes in the data or model can be easily retraced. It helps maintain the integrity of the ML pipeline, avoiding issues like "data drift," where changes in the data over time might affect model performance. With version control, models can be tested against previous versions of the data to understand how they perform under different conditions.

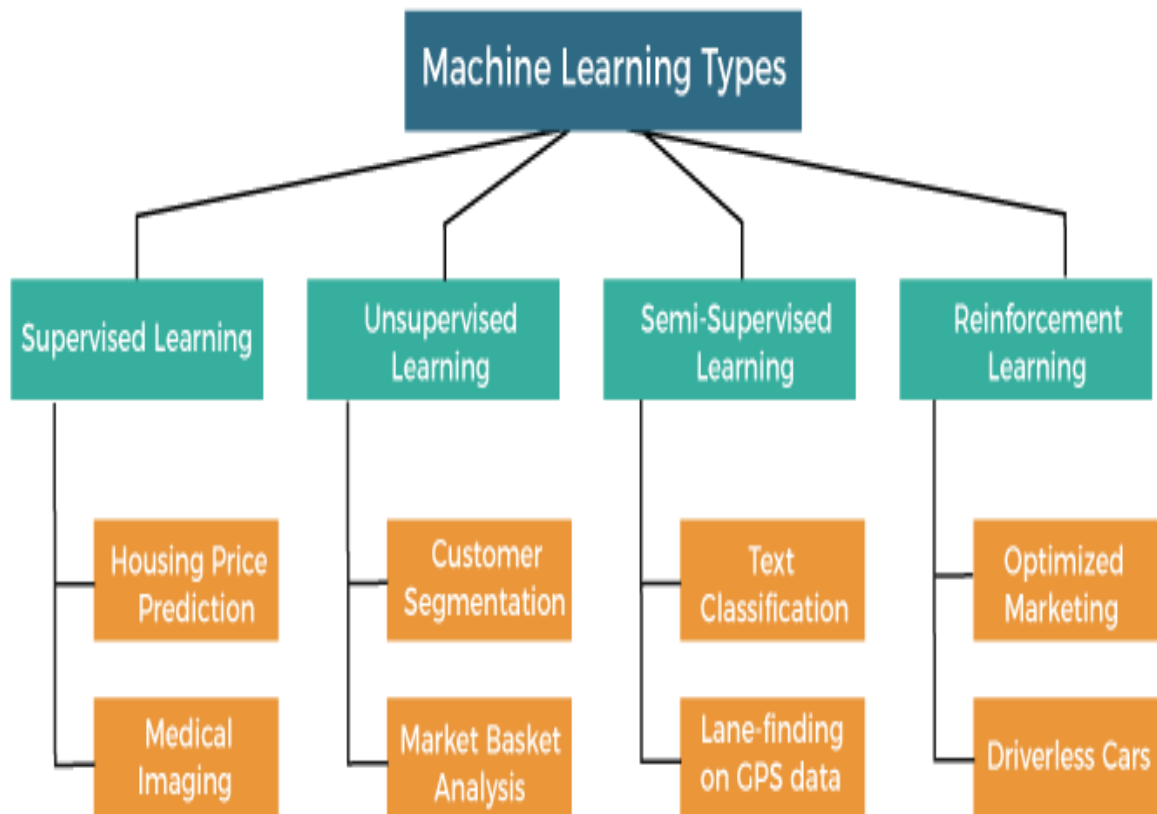
## Data Modeling

**Data modeling** is the process of organizing and structuring data to effectively represent relationships and patterns. In the context of machine learning, it refers to visualizing the data structures and the connections between different data points. The goal of data modeling is to present the data in a way that reveals insights into its relationships, distributions, and interdependencies. This step often involves creating conceptual, logical, or physical models that showcase how data will be represented and accessed in the system. Data models help inform the feature engineering process, where specific aspects of the data are selected or transformed to optimize model training. Good data modeling helps ensure that the data is well-aligned with the objectives of the ML project, providing a clear roadmap for the subsequent modeling stages.

## Model Training and Retraining

**Model training** involves feeding prepared data into an algorithm to build a predictive model. The model learns patterns from the training data and adjusts its parameters to minimize prediction errors. This process, known as **model fitting**, is essential for ensuring that the model can generalize well to unseen data. After the model is deployed, **model retraining** becomes crucial for maintaining its relevance. As new data is ingested or when underlying patterns in the data shift, continuous retraining helps adapt the model to these changes. This process can be automated within an MLOps pipeline to ensure that the model remains up-to-date and continues to provide accurate predictions. The retraining process typically involves evaluating model performance using validation data and adjusting hyperparameters to optimize accuracy over time.

### 3.2 Types of Machine Learning



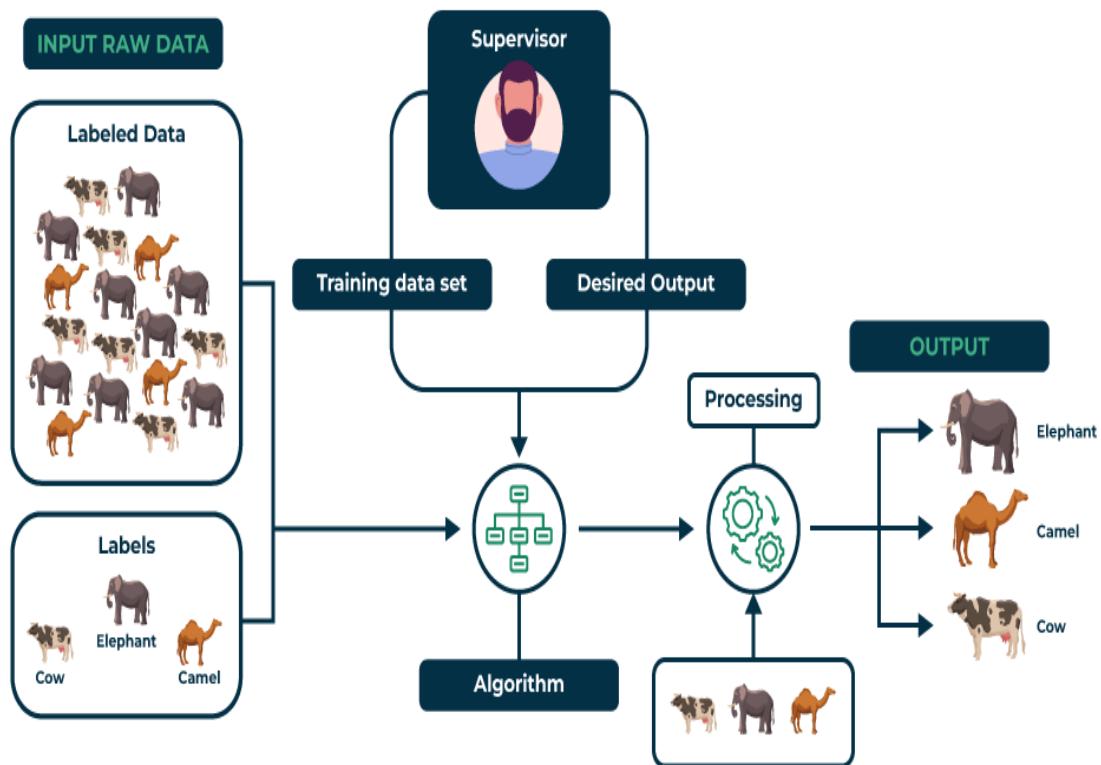
**Fig 3.2 Machine Learning Types**

#### 1. Supervised machine learning

Supervised machine learning is a fundamental approach where the model is trained on a labeled dataset, meaning the input data (features) and the corresponding output (target or label) are provided. In this process, the model learns to map input features to known outcomes by identifying patterns and relationships in the data.

For example, when building a tornado forecasting model, data scientists would use historical records of tornado events, with features such as date, location, temperature, wind speed, and atmospheric pressure. The target variable would be the actual tornado activity observed on those days, such as whether a tornado occurred, its severity, or its location. The model is then trained to predict the tornado activity based on these input features. Over time, by processing many labeled examples, the algorithm improves its ability to generalize to new, unseen data, making accurate predictions on future events. Supervised learning is commonly used in tasks such as classification (e.g., determining whether an email is spam or not) and regression (e.g., predicting house prices based on features like square footage, location, and age).

# Supervised Learning



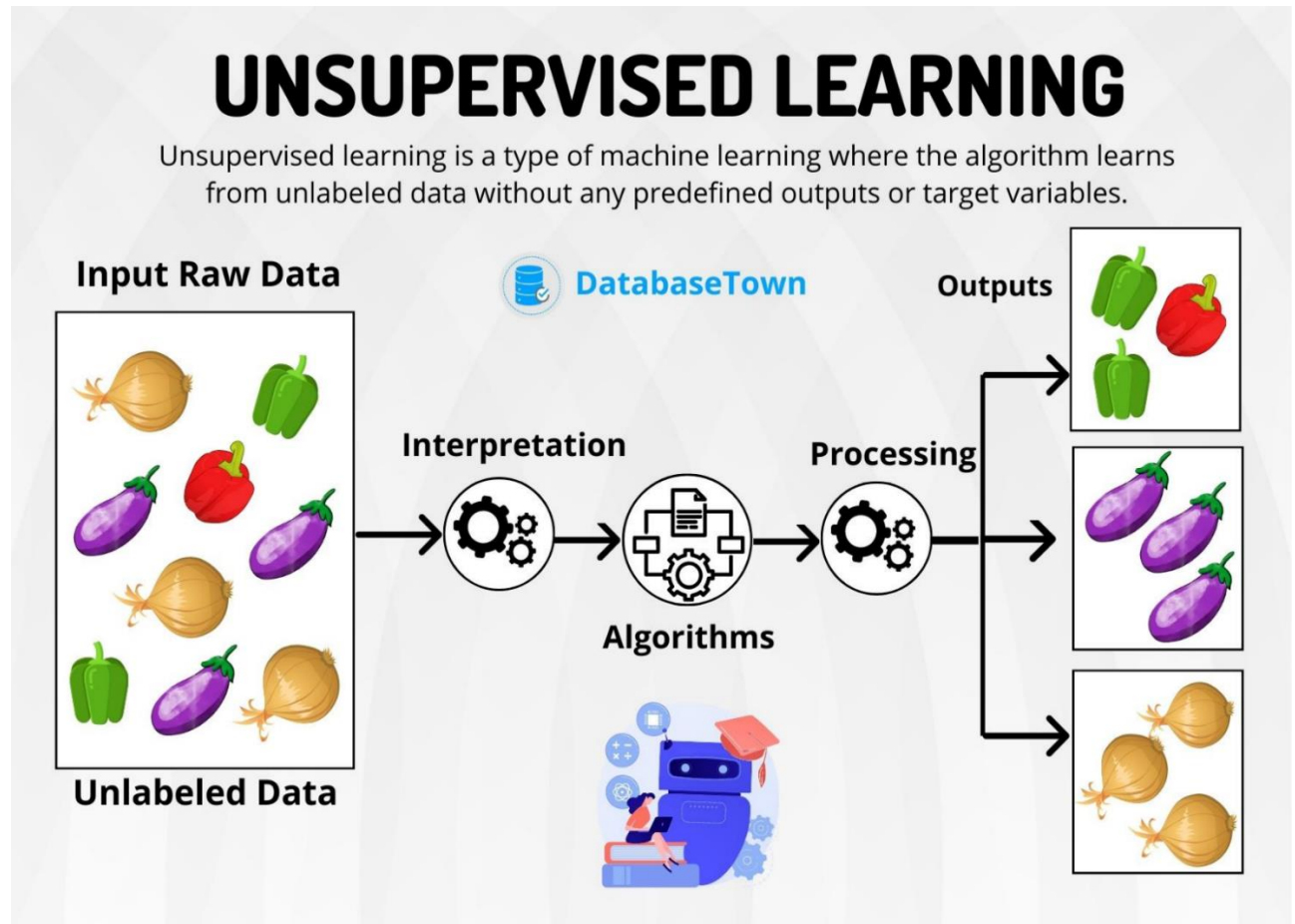
## 2. Unsupervised Machine Learning

Unsupervised machine learning involves training algorithms on datasets that do not have labels or predefined outputs. Instead, these algorithms aim to discover hidden structures or patterns in the data through techniques like clustering, dimensionality reduction, or association. For instance, algorithms like Apriori, Gaussian Mixture Models (GMMs), and Principal Component Analysis (PCA) are widely used in unsupervised learning. These methods allow data scientists to explore and interpret large datasets, identifying key trends or relationships that were not previously obvious. One of the most common unsupervised learning techniques is cluster analysis, which groups data points into clusters based on similarities in their features. For example, in customer segmentation, clustering algorithms can help businesses identify distinct groups of customers with similar behaviors, allowing for more personalized marketing strategies.

Similarly, anomaly detection uses clustering to identify unusual or rare data points, which is useful in fraud detection or network security. Additionally, association algorithms are valuable in discovering relationships between items within large databases. For example, in retail, association rules could reveal patterns like "customers who buy bread often also buy butter," enabling better inventory and marketing decisions. Unsupervised learning excels in situations where labeled data is scarce or when the goal is to uncover hidden insights in complex datasets.



The most common unsupervised learning method is cluster analysis, which uses clustering algorithms to categorize data points according to value similarity (as in customer segmentation or anomaly detection). Association algorithms allow data scientists to identify associations between data objects inside large databases, facilitating data visualization and dimensionality reduction.



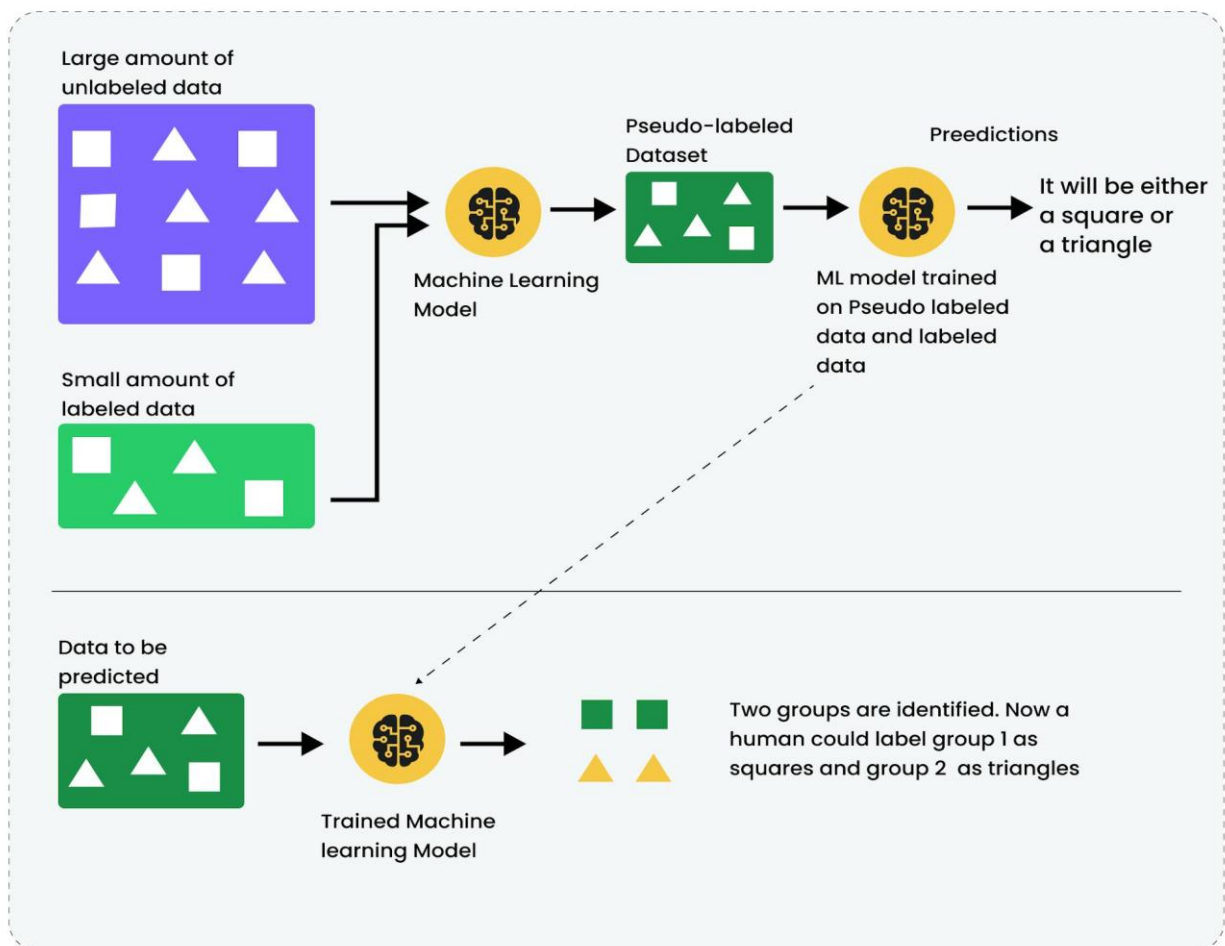
### 3.Semi-supervised learning

Semi-supervised learning is a hybrid machine learning approach that leverages both labeled and unlabeled data. In this model, a small amount of labeled data is used to guide the learning process for a larger pool of unlabeled data, allowing the algorithm to benefit from the vast amount of unlabeled data that would otherwise be unused. This approach strikes a balance between supervised learning, which requires a lot of labeled data, and unsupervised learning, which works without labels but may struggle with providing precise predictions.

Semi-supervised learning can be particularly useful in situations where labeling data is expensive or time-consuming, but unlabeled data is abundant. For instance, a semi-supervised learning model might first apply unsupervised learning techniques, such as clustering, to identify distinct groups within the unlabeled data. Then, the small labeled dataset can be used to assign specific labels to these groups.

One prominent example of semi-supervised learning is Generative Adversarial Networks (GANs). GANs use two neural networks—one for generating data and another for discriminating between real and generated data. Through this adversarial process, GANs can produce high-quality unlabeled data, which is then used to enhance the learning process. This type of learning is particularly useful in domains such as image recognition, where manually labeling large datasets would be impractical.

## Semi-supervised learning



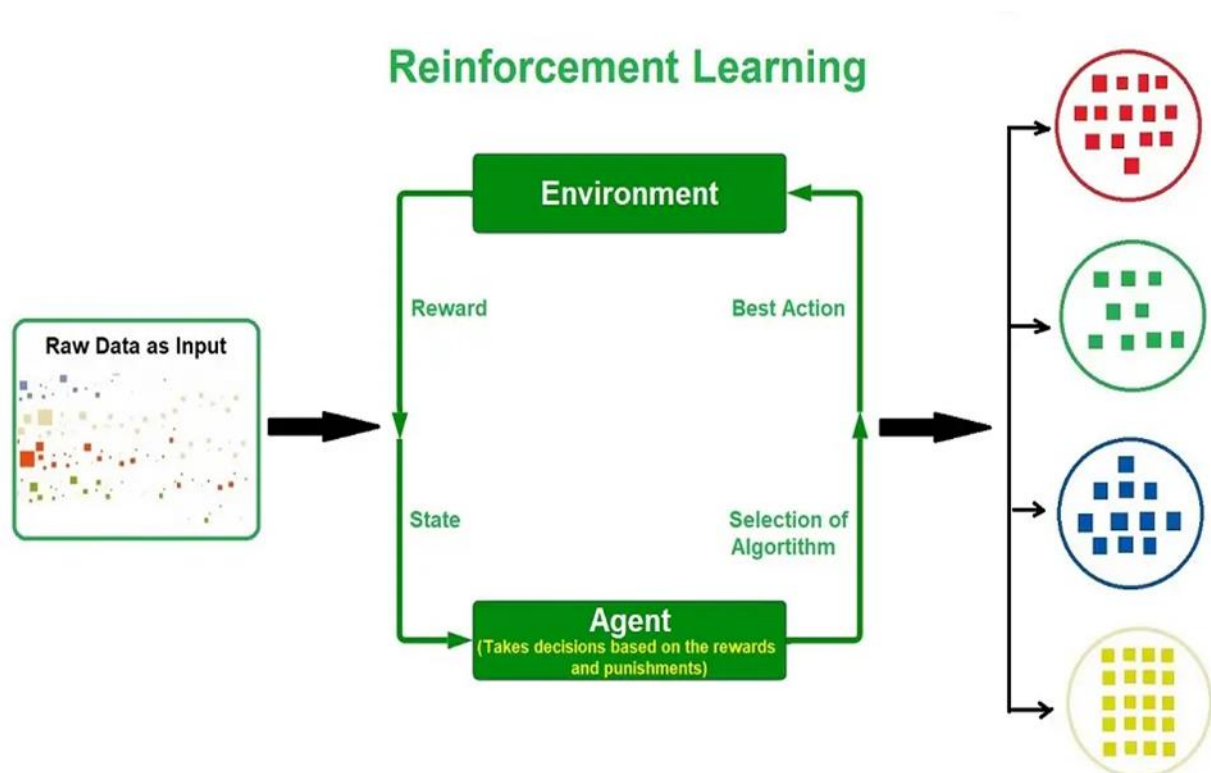
## 4.Reinforcement learning

Reinforcement learning (RL) is a unique type of machine learning based on the concept of an agent learning to make decisions by interacting with an environment. In RL, the agent takes actions within a defined environment, with the goal of maximizing a reward signal.

The agent receives feedback in the form of rewards or penalties based on the actions it takes. Positive feedback (rewards) encourages the agent to repeat actions that lead to favorable outcomes, while negative feedback (punishments) discourages the agent from repeating undesirable behaviors. This feedback loop enables the agent to learn from experience and adapt over time.

One key feature of reinforcement learning is that the agent does not require a labeled dataset; instead, it learns from trial and error in a dynamic setting. The goal is to find the optimal strategy or policy that maximizes the cumulative reward. RL is commonly used in complex decision-making problems where the solution is not obvious or deterministic. A common example is video game development, where RL algorithms train characters to play games autonomously, learning strategies like avoiding obstacles or defeating opponents by interacting with the game environment.

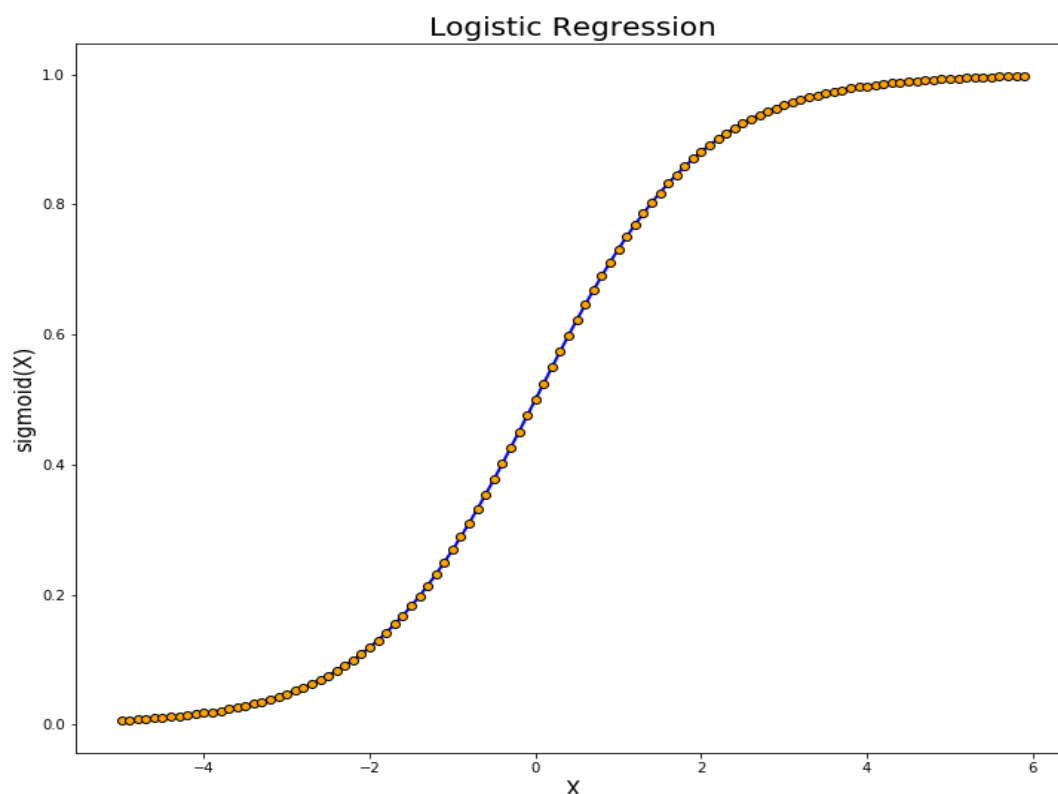
Similarly, RL has shown great promise in robotics, where robots are trained to perform tasks such as walking, grasping objects, or navigating complex environments. These tasks typically involve continuous learning and adaptation, making RL an ideal approach for modeling such dynamic behaviors. Through repeated interactions and feedback, RL algorithms can eventually discover optimal or near-optimal solutions that would be difficult to define explicitly



### 3.3 Machine Learning Algorithms

**Linear regression** : is a foundational algorithm in machine learning used to model the relationship between a dependent variable and one or more independent variables. By fitting a linear equation to the observed data, it predicts continuous outcomes, making it particularly useful in various domains such as finance, healthcare, and social sciences. The model aims to minimize the sum of squared differences between predicted and actual values, often achieved through Ordinary Least Squares (OLS). One of its strengths is interpretability, as the coefficients indicate the impact of each predictor on the outcome. However, linear regression assumes a linear relationship and is sensitive to outliers, which can affect its performance. Extensions like multiple linear regression and regularized forms such as Ridge and Lasso regression address some of these limitations, making linear regression a versatile and widely-used tool in predictive modeling.

**Logistic Regression**: is a Supervised algorithm which mostly works in the case of binary classification problems. Logistic regression is a sophisticated algorithm where the data to be trained using this algorithm should be properly presented i.e., Normalized/Scaled, Columns should be Converted to numerical and data should be neat and clean. The output is presented in the form of logit score, where this helps us to predict the likelihood of an event occurring of a given problem. The main reason of getting a S curve in the below chart is that the sigmoid function does the trick of converting the given number in the range between 0 and 1.



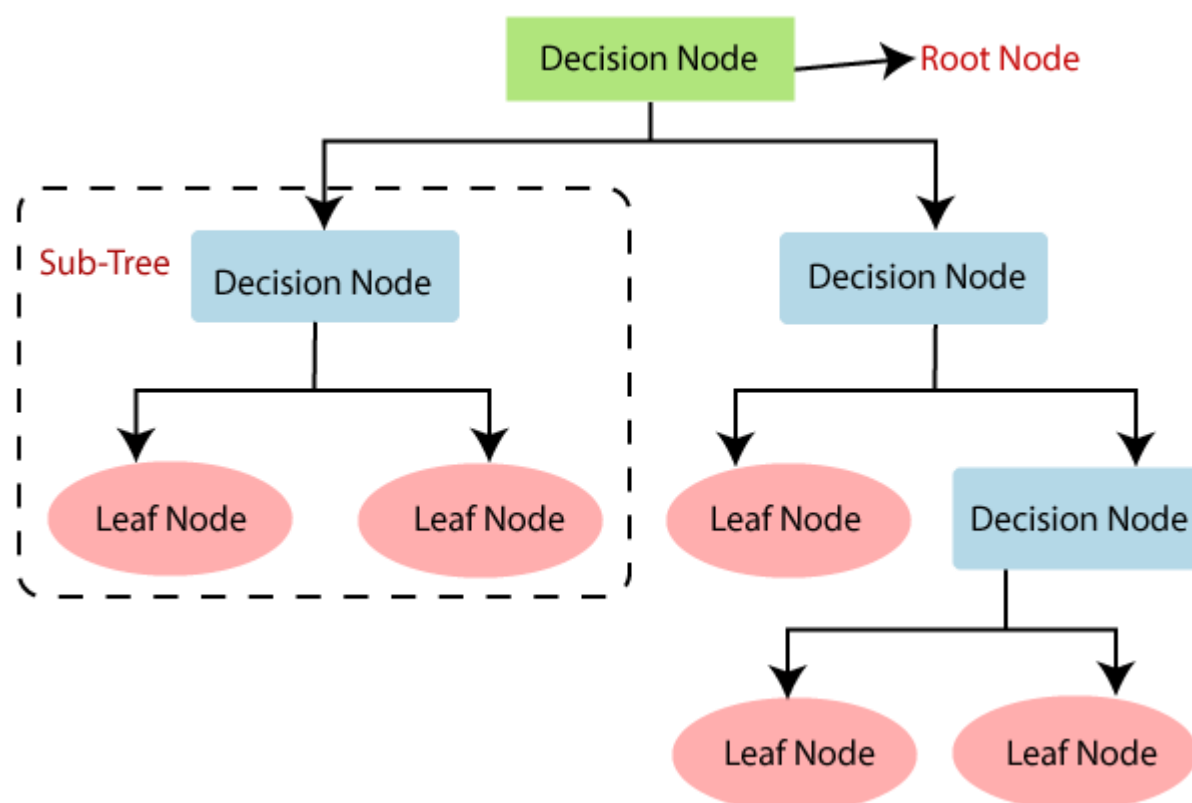
**Fig 3.2.1 Graph of Logistic Regression**

$$\text{Sigmoid}(x) = \frac{1}{1 + e^{-x}}$$

**Random Forest Classifier:** is an ensemble algorithm which works with multiple algorithms parallelly. This is a supervised algorithm and it can be used with both classification and regression problems. The output of the new data is estimated either by using majority voting or average voting technique. Since the algorithm works with bagging technique, multiple decision trees are used to provide the output for the specific input. This is a key difference between decision trees and random forests. While decision trees consider all the possible feature splits, random forests only select a subset of those features. Random forest works best with large datasets and high dimensional.

**Decision Tree Classifier:** is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome. It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions.

and the process for making decisions is in a tree like structure, decision tree is a supervised machine learning algorithm mainly used for predicting the outcome after computing all the attributes. The process flow of Decision tree goes from Root node to leave node i.e., the decision node.



**Fig 3.2.2 Decision Tree**

**Naïve Bayes** :is a family of probabilistic algorithms based on Bayes' theorem, primarily used for classification tasks. It operates under the "naïve" assumption that features are conditionally independent given the class label, which simplifies the computation of probabilities. This algorithm is particularly effective for text classification, such as spam detection and sentiment analysis, due to its ability to handle large datasets and its speed in training and prediction. The model calculates the posterior probability of each class by combining the prior probability of the class with the likelihood of the features, allowing it to make predictions based on the highest probability. Despite its simplicity, Naïve Bayes performs surprisingly well even when the independence assumption is violated. Its strengths include efficiency, scalability, and interpretability, making it a popular choice for many practical applications in natural language processing and beyond.

**Dimensionality reduction algorithms** :are techniques used to reduce the number of features in a dataset while preserving essential information. This is particularly valuable in high-dimensional spaces, where the "curse of dimensionality" can hinder model performance and increase computational costs. Common methods include Principal Component Analysis (PCA), which transforms data into a lower-dimensional space by identifying the directions of maximum variance, and t-Distributed Stochastic Neighbor Embedding (t-SNE), which is effective for visualizing high-dimensional data in two or three dimensions. Other techniques include Linear Discriminant Analysis (LDA) and Autoencoders. Dimensionality reduction not only improves computational efficiency but also helps in visualizing data, removing noise, and mitigating overfitting, ultimately leading to more robust models.

**Gradient Boosting** : is a powerful ensemble learning technique used for both classification and regression tasks. It builds models sequentially, where each new model attempts to correct the errors made by the previous ones. The core idea is to combine weak learners, typically decision trees, into a strong predictive model by optimizing a loss function through gradient descent. Each tree is trained on the residual errors of the combined predictions from previous trees, allowing the algorithm to focus on difficult-to-predict instances. Gradient Boosting is known for its high accuracy and flexibility, with popular implementations like XGBoost, LightGBM, and CatBoost that introduce enhancements such as regularization and efficient handling of large datasets. However, it can be prone to overfitting if not carefully tuned. Overall, Gradient Boosting is widely used in machine learning competitions and real-world applications due to its robustness and performance.

**The k-nearest neighbors algorithm, or KNN** : is a non-parametric, supervised learning method. It classifies or predicts the grouping of a data point based on its proximity to neighboring points. KNN is a versatile tool widely used in machine learning for various classification and regression tasks. The abbreviation KNN stands for "K-Nearest Neighbour". It is a supervised machine learning algorithm. The algorithm can be used to solve both classification and regression problem statements. The number of nearest neighbours to a new unknown variable that has to be predicted or classified is denoted by the symbol 'K'.

**K-means** : is an unsupervised learning method for clustering data points. The algorithm iteratively divides data points into K clusters by minimizing the variance in each cluster.,is an iterative algorithm that splits a dataset into non-overlapping subgroups that are called clusters. The amount of clusters created is determined by the value of k – a hyperparameter that's chosen before running the algorithm.



**Fig 3.2.3 K-Means graph**

## 4 DATASET DESCRIPTION

The Dataset is collected from Kaggle Repository which contains 5435 Instances with 21 features. Some of the features which has a very high correlation with our target variable i.e., Covid(Yes/No) are Breathing, Fever, cold, cough, headache, Fatigue, abroad travel etc., The dataset requires data balancing since one of the classes are very low compared to the other class, which might result in data bias. There aren't any null values and data cleaning is perfect which makes ready to eat data from the people who would like to work on it. Notable features include symptoms such as breathing difficulties, fever, cold, cough, headache, and fatigue, as well as travel history abroad. This correlation can provide valuable insights into symptom patterns and help identify high-risk individuals.

This dataset is devoid of null values, which simplifies the preprocessing stage. Clean data is crucial for building reliable models, as missing or inaccurate data can lead to incorrect conclusions and affect the overall quality of the analysis. The absence of null values suggests that the dataset has undergone a thorough data collection process, ensuring that the responses are complete and ready for analysis.

This quality makes the dataset particularly appealing to researchers and data scientists looking to explore COVID-19 correlations without the need for extensive cleaning. In addition to its readiness for analysis, the dataset's comprehensive nature allows for various types of statistical analyses and machine learning applications. Researchers can perform exploratory data analysis (EDA) to visualize relationships between features and the target variable. This step can help identify patterns and guide feature selection for predictive modeling. Moreover, different machine learning algorithms—ranging from logistic regression to more complex models like random forests or neural networks can be employed to predict likelihood of COVID-19 infection based on the available features.

Finally, the potential implications of analyzing this dataset are significant. By understanding which symptoms are most indicative of COVID-19, healthcare providers can better screen individuals and allocate resources effectively. Moreover, findings from such analyses can inform public health policies and guidelines, helping to prevent the spread of the virus. Overall, this dataset represents a valuable asset for advancing research.

Breathing Problem	Fever	Dry Cough	Sore throat	Running Nose	Asthma	Chronic Lung Disease	Headache	Heart Disease	Diabetes	Hyper Tension	Fatigue	Gastrointestinal	Abroad travel
Yes	Yes	Yes	Yes	Yes	No	No	No	No	Yes	Yes	Yes	Yes	No
Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	No	No	No	Yes	No	No
Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	No	Yes	Yes	Yes
Yes	Yes	Yes	No	No	Yes	No	No	Yes	Yes	No	No	No	Yes
Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	No	Yes	No
Yes	Yes	Yes	No	No	No	No	No	Yes	No	Yes	No	No	No
Yes	Yes	Yes	No	No	No	Yes	No	Yes	Yes	Yes	Yes	Yes	No

**Table 4.1 Dataset**



Contact with COVID Patient	Attended Large Gathering	Visited Public Exposed Places	Family working in Public Exposed Places	Wearing Masks	Sanitization from Market	COVID-19
Yes	No	Yes	Yes	No	No	Yes
No	Yes	Yes	No	No	No	Yes
No	No	No	No	No	No	Yes
No	Yes	Yes	No	No	No	Yes
Yes	No	Yes	No	No	No	Yes
No	No	No	No	No	No	Yes
No	Yes	Yes	Yes	No	No	Yes

**Table 4.2 Dataset**

## Issues Faced:

**Manual Installation of Specific Libraries** Occasionally, while working with Python projects, certain libraries may not install automatically due to compatibility issues or specific version requirements. In these cases, it's necessary to manually install the required libraries using `pip install <module name>`. For example, you can use `pip install pandas` to install the pandas library. Additionally, ensure that you're using the correct version of pip or pip3 as per your Python installation.

**Python Version Compatibility** Using the latest version of Python or a specific version compatible with your project is essential to avoid compatibility issues. Certain libraries and frameworks may only work with specific versions of Python, so it's important to verify version requirements before starting. You can check the version of Python by running `python --version` or `python3 --version` in your terminal.

**Setting Environment Variables for Python and Anaconda** To execute Python scripts and use the Anaconda environment smoothly across different code editors (e.g., VS Code, PyCharm), it's crucial to set up the Python path in your system's environment variables. This configuration allows you to run Python files from any location in the command line without specifying the full path.

**Updating CodePaths for Datasets and Models** When working with datasets or pre-trained models in your project, make sure to adjust any hardcoded paths in your code to match the actual directory locations on your system. This ensures that your scripts can locate the files and resources they need without causing errors due to incorrect paths.

**Virtual Environment Setup:** Using virtual environments for each project helps to manage dependencies independently and prevent version conflicts between projects. You can create a virtual environment using `python -m venv <env_name>` and activate it with the appropriate command based on your OS (e.g., `source <env_name>/bin/activate` for Mac/Linux or `<env_name>\Scripts\activate` for Windows).

**Updating Pip and Installed Packages Regularly:** To keep all libraries and packages in your project up-to-date, use the command `pip install --upgrade pip` to update pip itself and `pip install --upgrade <library_name>` for updating specific libraries. This can help avoid compatibility issues with newer Python versions and ensures your libraries have the latest features and security patches.

**Installing Requirements from a Requirements File:** If your project requires multiple libraries, you can save all dependencies in a `requirements.txt` file, which can be shared and installed using pip

install -r requirements.txt. This is especially helpful for collaborators, as it provides a clear list of all packages needed to run the project without manual installation.

**Configuring Python Interpreters in Code Editors:** Most code editors allow you to select a specific Python interpreter to use, which can be particularly useful when switching between projects with different Python versions or virtual environments. Make sure to configure the correct interpreter in your editor's settings (e.g., in VS Code, select the interpreter via the Command Palette).

**Troubleshooting Common Import Errors:** Import errors can often arise when a library is installed in a different Python environment or due to path issues. Running `pip list` can confirm if a package is installed, and checking the `PYTHONPATH` variable can help in identifying if your system paths are set correctly. Reinstalling the library can also sometimes fix these issues.

**Setting Up Jupyter Notebook or IDE Kernels:** For Jupyter Notebook or JupyterLab, ensure that the correct kernel is selected to use the desired Python environment. You can add a virtual environment as a Jupyter kernel by installing `ipykernel` and running `python -m ipykernel install --user --name <env_name>`. This ensures that the notebook operates within the specified environment, allowing consistent library access.

## 5 IMPLEMENTATION AND TESTING

Before going to implementation of the model there are two important concepts those are

1. Evaluation Metrics
2. Confusion Matrix
3. Cross Validation
4. Train/Test Split
5. Learning Curves
6. Hyperparameter Tuning
7. Feature Importance

### 1. Evaluation metrics

are a crucial aspect of any machine learning or deep learning project, as they enable practitioners to assess how well their models perform, particularly when faced with new or unseen data. These metrics provide insights into the model's effectiveness and robustness, allowing data scientists to make informed decisions about model improvements and deployment. Various evaluation metrics are available, each tailored for specific types of problems and datasets. For instance, metrics like the ROC AUC curve, F1 score, recall, and precision serve different purposes, and selecting the appropriate metric is essential for accurately gauging model performance.

In our project, we have opted to use the confusion matrix and classification report as our primary evaluation tools. The confusion matrix offers a visual representation of the model's predictions compared to the actual outcomes, detailing true positives, true negatives, false positives, and false negatives. This matrix serves as the foundation for deriving several critical metrics, including accuracy, precision, recall, and the F1 score. By analyzing these metrics, we can gain a comprehensive understanding of the model's strengths and weaknesses, ultimately guiding us toward more effective strategies for enhancement.

The classification report complements the confusion matrix by providing a summary of the key metrics in a structured format. It presents precision, recall, and F1 scores for each class, making it easier to interpret model performance across different categories. This detailed breakdown is particularly valuable in multi-class classification problems, where some classes may be more challenging to predict accurately than others. By leveraging both the confusion matrix and classification report, we can thoroughly evaluate our model's performance, identify areas for improvement, and ensure that it meets the desired performance standards before deployment in real-world applications.

### 2. Confusion Matrix

a visualization tool that compares a model's predicted values to the actual values of a dataset, and is used to evaluate the performance of a classification model. The matrix is structured with rows representing the actual classes and columns representing the predicted classes, allowing for a straightforward comparison of model predictions against the true outcomes.

**Classification Report:** Classification report helps us to understand and evaluate how good the model is performing which consists of different evaluation metrics, such as

## Confusion Matrix

	Actually Positive (1)	Actually Negative (0)
Predicted Positive (1)	True Positives (TPs)	False Positives (FPs)
Predicted Negative (0)	False Negatives (FNs)	True Negatives (TNs)

**Key Components:** It includes four primary elements:

- **True Positives (TP):** Correctly predicted positive instances.
- **True Negatives (TN):** Correctly predicted negative instances.
- **False Positives (FP):** Incorrectly predicted positive instances (Type I error).
- **False Negatives (FN):** Incorrectly predicted negative instances (Type II error).

**Performance Metrics:** From the confusion matrix, various performance metrics can be calculated, such as:

- **Accuracy:** The proportion of total correct predictions  $(TP + TN) / \text{Total predictions}$ .
- **Precision:** The ratio of true positives to the total predicted positives  $(TP / (TP + FP))$ , indicating the model's accuracy in positive predictions.
- **Recall (Sensitivity):** The ratio of true positives to the total actual positives  $(TP / (TP + FN))$ , reflecting the model's ability to identify positive instances.
- **F1 Score:** The harmonic mean of precision and recall, providing a balanced measure  $(F1 \text{ Score} = 2 * (\text{Precision} * \text{Recall}) / (\text{Precision} + \text{Recall}))$ .

### 3. Cross-Validation:

Cross-validation is a robust statistical method used to evaluate the performance of a machine learning model. One of the most common techniques is k-fold cross-validation, where the dataset is divided into k equally sized folds. The model is trained on k-1 folds and validated on the remaining fold. This process is repeated k times, with each fold serving as the validation set once. The results from each fold are then averaged to provide a more reliable estimate of the model's performance. This approach mitigates the risk of overfitting to a single training set and offers a better understanding of how the model will perform on unseen data.

Moreover, cross-validation helps identify how sensitive a model is to the specific data it was trained on. If the performance varies significantly across different folds, it may indicate issues like overfitting or that the model is not capturing the underlying patterns well. By using cross-validation, practitioners can ensure that the model generalizes better, providing a more consistent and reliable performance evaluation, which is crucial when making decisions based on the model's predictions.

#### **4. Train/Test Split:**

The train/test split is a fundamental technique for evaluating a machine learning model's ability to generalize to new, unseen data. Typically, a dataset is divided into two parts: the training set, which is used to train the model, and the testing set, which is reserved for evaluating the model's performance after training. A common split ratio is 70/30 or 80/20, depending on the size of the dataset and the complexity of the model. The training set allows the model to learn patterns, while the test set provides an unbiased evaluation of its performance.

This method ensures that the model's performance metrics are reflective of its ability to make accurate predictions on real-world data. If a model performs exceptionally well on the training set but poorly on the test set, it may indicate overfitting, where the model has memorized the training data rather than learning to generalize from it. Thus, a proper train/test split is critical for assessing a model's effectiveness and making informed decisions about its deployment in practical applications.

#### **5. Learning Curves:**

Learning curves are a powerful tool for diagnosing the behavior of machine learning models. By plotting the training and validation performance against varying sizes of the training dataset, one can visualize how the model learns over time. These curves can reveal crucial insights about bias and variance in the model. If the training performance is significantly better than the validation performance, this may suggest overfitting, indicating that the model is capturing noise rather than the underlying data distribution.

Conversely, if both training and validation performances are low, the model may be underfitting, meaning it is too simplistic to capture the complexity of the data. Analyzing learning curves helps practitioners understand the learning dynamics and make necessary adjustments to the model or feature engineering process. For example, if the validation performance improves significantly with more training data, it may be beneficial to gather more data or refine the model architecture.

#### **6. Hyperparameter Tuning:**

Hyperparameter tuning is a critical step in optimizing the performance of a machine learning model. Hyperparameters are settings that are not learned from the data but are set prior to the training process, such as learning rate, number of trees in a random forest, or regularization strength. Finding the right combination of hyperparameters can significantly enhance model performance. Techniques like grid search and random search are commonly used for this purpose. Grid search exhaustively evaluates a specified subset of hyperparameters, while random search samples a range of hyperparameter combinations, which can be more efficient in exploring the hyperparameter space.

The process of tuning hyperparameters is crucial because even small adjustments can lead to significant changes in model accuracy and generalization. Tools such as cross-validation are often integrated into hyperparameter tuning to ensure that the selected parameters provide robust performance across different subsets of the data. Ultimately, effective hyperparameter tuning helps achieve the best possible model performance, ensuring it meets the specific requirements of the task at hand.

## 7. Feature Importance:

Analyzing feature importance is a vital aspect of understanding a machine learning model's behavior and performance. Feature importance metrics indicate which features (or variables) have the most significant impact on the model's predictions. This analysis can be performed using various methods, including tree-based models that naturally provide importance scores or techniques like permutation importance that evaluate the effect of removing or permuting features on the model's accuracy. Understanding feature importance helps in identifying key drivers of the predictions and can guide further feature selection and engineering efforts.

Moreover, analyzing feature importance can lead to improved model interpretability, which is particularly important in applications where decisions based on model predictions can have significant consequences, such as healthcare or finance. By focusing on the most impactful features, practitioners can simplify their models, reduce overfitting, and increase efficiency without sacrificing performance. Additionally, this insight can inform stakeholders about the underlying factors influencing predictions, ultimately leading to more transparent and trustworthy machine learning applications.

Implementation of Model:

```
covid=pd.read_csv(r'/content/sample_data/Covid Dataset.csv')
```

```
features = covid.columns.tolist()
```

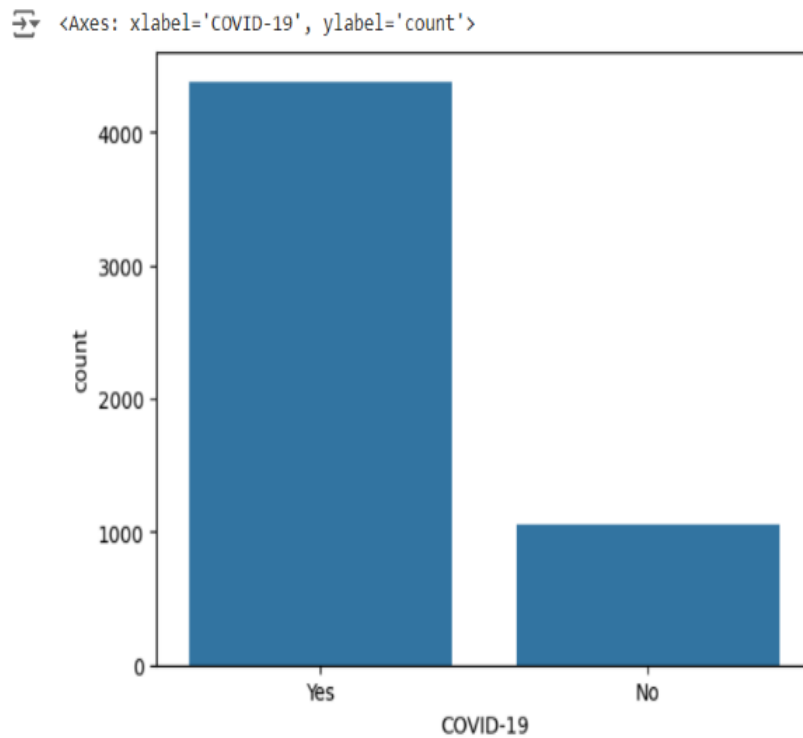
```
print(features)
```

data bias. There aren't any null values and data cleaning is perfect which makes ready to eat data from the people who would like to work on it.

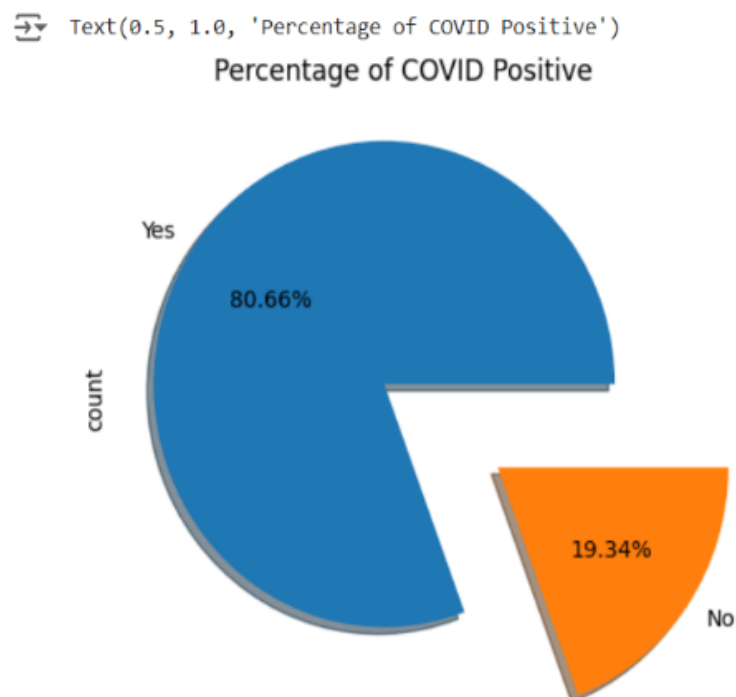
Output:

```
['Breathing Problem', 'Fever', 'Dry Cough', 'Sore throat', 'Running Nose', 'Asthma', 'Chronic Lung Disease', 'Headache', 'Heart Disease', 'Diabetes', 'Hyper Tension', 'Fatigue ', 'Gastrointestinal ', 'Abroad travel', 'Contact with COVID Patient', 'Attended Large Gathering', 'Visited Public Exposed Places', 'Family working in Public Exposed Places', 'Wearing Masks', 'Sanitization from Market']
```

```
sns.countplot(x='COVID-19',data=covid)
```



```
covid["COVID19"].value_counts().plot.pie(explode=[0.1,0.5],autopct='%1.2f%',shadow=True)  
plt.title('Percentage of COVID Positive')
```



```

from imblearn.over_sampling import RandomOverSampler
import pandas as pd
data=pd.DataFrame({})
x=cov.drop('COVID-19',axis=1)
y=cov['COVID-19']
correlation=cov.corr()
correlation.style.background_gradient(cmap='PiYG',axis=None)

```

	Breathing Problem	Fever	Dry Cough	Sore throat	Running Nose	Asthma	Chronic Lung Disease	Headache	Heart Disease	Diabetes	Hyper Tension	Fatigue	Gastrointestinal	Abroad travel	Contact with COVID Patient
Breathing Problem	1.000000	0.089903	0.159562	0.303768	0.055190	0.075318	-0.098291	-0.062172	-0.073366	0.055427	0.045256	0.000561	-0.075390	0.117795	0.214634
Fever	0.089903	1.000000	0.127580	0.322235	0.081758	0.073953	-0.025160	-0.035416	-0.031462	0.050286	0.079001	-0.060458	-0.008067	0.128726	0.164704
Dry Cough	0.159562	0.127580	1.000000	0.213907	-0.030763	0.086843	-0.043664	-0.035912	0.047566	-0.006593	0.081989	-0.039909	0.008251	0.331418	0.128330
Sore throat	0.303768	0.322235	0.213907	1.000000	0.039450	0.081377	-0.050440	-0.015971	0.002177	0.001938	0.042811	-0.023290	0.025886	0.205986	0.189251
Running Nose	0.055190	0.081758	-0.030763	0.039450	1.000000	-0.022763	-0.014376	0.068479	-0.056750	0.042961	-0.020445	0.007026	-0.014673	0.034526	0.003776
Asthma	0.075318	0.073953	0.086843	0.081377	-0.022763	1.000000	-0.033771	0.037064	0.078783	-0.012060	0.017707	0.006564	0.101909	0.068286	0.005046
Chronic Lung Disease	-0.098291	-0.025160	-0.043664	-0.050440	-0.014376	-0.033771	1.000000	-0.050480	-0.039860	0.046789	-0.010331	-0.047655	-0.050333	-0.088854	-0.062482
Headache	-0.062172	-0.035416	-0.035912	-0.015971	0.068479	0.037064	-0.050480	1.000000	0.048471	0.032390	-0.207489	0.052035	0.097778	0.043589	-0.082101
Heart Disease	-0.073366	-0.031462	0.047566	0.002177	-0.056750	0.076783	-0.039860	0.048471	1.000000	-0.032956	0.049139	-0.058925	0.004121	-0.020761	-0.025593
Diabetes	0.055427	0.050286	-0.006593	0.001938	0.042961	-0.012060	0.046789	0.032390	-0.032956	1.000000	0.042543	-0.043903	0.040651	0.039013	-0.085696
Hyper Tension	0.045256	0.079001	0.081989	0.042811	-0.020445	0.017707	-0.010331	-0.207489	0.049139	0.042543	1.000000	-0.027605	-0.067972	-0.016382	0.027307
Fatigue	0.000561	-0.060458	-0.039909	-0.023290	0.007026	0.006564	-0.047655	0.052035	-0.058925	-0.043903	-0.027605	1.000000	0.009356	-0.068401	-0.027383

```


covid['Breathing Problem']=e.fit_transform(covid['Breathing Problem'])
covid['Fever']=e.fit_transform(covid['Fever'])
covid['Dry Cough']=e.fit_transform(covid['Dry Cough'])
covid['Sore throat']=e.fit_transform(covid['Sore throat'])
covid['Running Nose']=e.fit_transform(covid['Running Nose'])
covid['Asthma']=e.fit_transform(covid['Asthma'])
covid['Chronic Lung Disease']=e.fit_transform(covid['Chronic Lung Disease'])
covid['Headache']=e.fit_transform(covid['Headache'])
covid['Heart Disease']=e.fit_transform(covid['Heart Disease'])
covid['Diabetes']=e.fit_transform(covid['Diabetes'])
covid['Hyper Tension']=e.fit_transform(covid['Hyper Tension'])
covid['Abroad travel']=e.fit_transform(covid['Abroad travel'])
covid['Contact with COVID Patient']=e.fit_transform(covid['Contact with COVID Patient'])
covid['Attended Large Gathering']=e.fit_transform(covid['Attended Large Gathering'])
covid['Visited Public Exposed Places']=e.fit_transform(covid['Visited Public Exposed Places'])

```



```
covid['Family working in Public Exposed Places']=e.fit_transform(covid['Family
working in Public Exposed Places'])
covid['Wearing Masks']=e.fit_transform(covid['Wearing Masks'])
covid['Sanitization from Market']=e.fit_transform(covid['Sanitization from
Market'])
covid['COVID-19']=e.fit_transform(covid['COVID-19'])
covid['Dry Cough']=e.fit_transform(covid['Dry Cough'])
covid['Sore throat']=e.fit_transform(covid['Sore throat'])
covid['Gastrointestinal ']=e.fit_transform(covid['Gastrointestinal '])
covid['Fatigue ']=e.fit_transform(covid['Fatigue '])
covid.head()
#before modifying data with LabelEncoder
```

[ ] covid.isnull()



	Breathing Problem	Fever	Dry Cough	Sore throat	Running Nose	Asthma	Chronic Lung Disease	Headache	Heart Disease	Diabetes	...	Fatigue	Gastrointestinal	Abroad travel	Contact with COVID Patient	Attended Large Gathering	Visited Public Exposed Places	Family working in Public Exposed Places	Wearing Masks
0	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
5429	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False
5430	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False
5431	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False
5432	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False
5433	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False

5434 rows x 21 columns

```
from sklearn.preprocessing import LabelEncoder
e=LabelEncoder()
#before modifying data with LabelEncoder
```



	Breathing Problem	Fever	Dry Cough	Sore throat	Running Nose	Asthma	Chronic Lung Disease	Headache	Heart Disease	Diabetes	...	Fatigue	Gastrointestinal	Abroad travel	Contact with COVID Patient	Attended Large Gathering	Visited Public Exposed Places	Family working in Public Exposed Places	Wearing Masks
0	1	1	1	1	1	0	0	0	0	1	...	1	1	0	1	0	1	1	0
1	1	1	1	1	0	1	1	1	0	0	...	1	0	0	0	1	1	0	0
2	1	1	1	1	1	1	1	1	0	1	...	1	1	1	0	0	0	0	0
3	1	1	1	0	0	1	0	0	1	1	...	0	0	1	0	1	1	0	0
4	1	1	1	1	1	0	1	1	1	1	...	0	1	0	1	0	1	0	0

5 rows x 21 columns

```
oversample=RandomOverSampler(sampling_strategy='auto',random_state=42)
X_resampled,y_resampled=oversample.fit_resample(x,y)
```

```

resampled_data=pd.DataFrame(X_resampled,columns=['feature1','feature2'])
resampled_data['label']=y_resampled

X_train, X_test, y_train, y_test = train_test_split(x, y, test_size = 0.25,
random_state=0)
def print_score(clf, X_train, y_train, X_test, y_test, train=True):
    if train:pred = clf.predict(X_train)
        clf_report = pd.DataFrame(classification_report(y_train, pred,output
dict=True
        print("Train
Result:\n=====")
        print(f"Accuracy Score: {accuracy_score(y_train, pred) * 100:.2f}%")
        print("_____")
        print(f"CLASSIFICATION REPORT:\n{clf_report}")
        print("_____")
        print(f"Confusion Matrix: \n {confusion_matrix(y_train, pred)}\n")

    elif train==False:
        pred = clf.predict(X_test)
        clf_report = pd.DataFrame(classification_report(y_test, pred,
output_dict=True))
        print("Test
Result:\n=====")
        print(f"Accuracy Score: {accuracy_score(y_test, pred) * 100:.2f}%")
        print("_____")
        print(f"CLASSIFICATION REPORT:\n{clf_report}")
        print("_____")
        print(f"Confusion Matrix: \n {confusion_matrix(y_test, pred)}\n")
lr_clf = LogisticRegression()
lr_clf.fit(X_train, y_train)
print_score(lr_clf, X_train, y_train, X_test, y_test, train=True)
print_score(lr_clf, X_train, y_train, X_test, y_test, train=False)

dt_clf = DecisionTreeClassifier()
dt_clf.fit(X_train, y_train)
print_score(dt_clf, X_train, y_train, X_test, y_test, train=True)
print_score(dt_clf, X_train, y_train, X_test, y_test, train=False)

rf_clf = RandomForestClassifier()
rf_clf.fit(X_train, y_train)
print_score(rf_clf, X_train, y_train, X_test, y_test, train=True)
print_score(rf_clf, X_train, y_train, X_test, y_test, train=False)

xgb_clf = GradientBoostingClassifier(random_state = 12345)
xgb_clf.fit(X_train, y_train)
print_score(xgb_clf, X_train, y_train, X_test, y_test, train=True)

```

```
print_score(xgb_clf, X_train, y_train, X_test, y_test, train=False)
```

```
new = X_test.iloc[176]
```

```
a = np.asarray(new)
```

```
a = a.reshape(1,-1)
```

```
p = lr_clf.predict(a)
```

```
[ ] if (p[0] == 1):  
    print("Person is affected by Covid 19 and is at risk of dying")  
else:  
    print("Great! the results are negative and you don't have to worry")
```

➡ Person is affected by Covid 19 and is at risk of dying

```
[ ] if (p[0] == 39):  
    print("Person is affected by Covid 19 and is at risk of dying")  
else:  
    print("Great! the results are negative and you don't have to worry")
```

➡ Great! the results are negative and you don't have to worry

## 6 RESULTS AND FINDING

Train Results for model in Logistic regression

Train Result:

---

---

Accuracy Score: 96.52%

---

CLASSIFICATION REPORT:

	0	1	accuracy	macro avg	weighted avg
precision	0.893204	0.983390	0.965153	0.938297	0.965906
recall	0.931646	0.973212	0.965153	0.952429	0.965153
f1-score	0.912020	0.978274	0.965153	0.945147	0.965430
support	790.000000	3285.000000	0.965153	4075.000000	4075.000000

---

Confusion Matrix:

```
[[ 736   54]
 [  88 3197]]
```

Test Result:

---

---

Accuracy Score: 96.32%

---

CLASSIFICATION REPORT:

	0	1	accuracy	macro avg	weighted avg
precision	0.895131	0.979853	0.963208	0.937492	0.963582
recall	0.915709	0.974499	0.963208	0.945104	0.963208
f1-score	0.905303	0.977169	0.963208	0.941236	0.963367
support	261.000000	1098.000000	0.963208	1359.000000	1359.000000

---

Confusion Matrix:

```
[[ 239   22]
 [  28 1070]]
```

## Train results for model in Decision Tree

### Train Result:

Accuracy Score: 97.57%

#### CLASSIFICATION REPORT:

	0	1	accuracy	macro avg	weighted avg
precision	0.939009	0.984489	0.975706	0.961749	0.975672
recall	0.935443	0.985388	0.975706	0.960416	0.975706
f1-score	0.937223	0.984938	0.975706	0.961080	0.975688
support	790.000000	3285.000000	0.975706	4075.000000	4075.000000

#### Confusion Matrix:

```
[[ 739  51]
 [ 48 3237]]
```

### Test Result:

Accuracy Score: 97.13%

#### CLASSIFICATION REPORT:

	0	1	accuracy	macro avg	weighted avg
precision	0.926923	0.981802	0.971302	0.954362	0.971262
recall	0.923372	0.982696	0.971302	0.953034	0.971302
f1-score	0.925144	0.982249	0.971302	0.953696	0.971281
support	261.000000	1098.000000	0.971302	1359.000000	1359.000000

#### Confusion Matrix:

```
[[ 241  20]
 [ 19 1079]]
```

Train results for model in Random Forest Classifier:

### Train Result:

Accuracy Score: 97.57%

#### CLASSIFICATION REPORT:

	0	1	accuracy	macro avg	weighted avg
precision	0.955204	0.980398	0.975706	0.967801	0.975514
recall	0.917722	0.989650	0.975706	0.953686	0.975706
f1-score	0.936088	0.985002	0.975706	0.960545	0.975519
support	790.000000	3285.000000	0.975706	4075.000000	4075.000000

#### Confusion Matrix:

```
[[ 725  65]
 [  34 3251]]
```

### Test Result:

Accuracy Score: 97.42%

#### CLASSIFICATION REPORT:

	0	1	accuracy	macro avg	weighted avg
precision	0.948413	0.980126	0.974246	0.964270	0.974036
recall	0.915709	0.988160	0.974246	0.951935	0.974246
f1-score	0.931774	0.984127	0.974246	0.957950	0.974072
support	261.000000	1098.000000	0.974246	1359.000000	1359.000000

#### Confusion Matrix:

```
[[ 239  22]
 [  13 1085]]
```

Train results for model in Gradient Boost:

Train Result:

Accuracy Score: 97.33%

CLASSIFICATION REPORT:

	0	1	accuracy	macro avg	weighted avg
precision	0.934866	0.982382	0.973252	0.958624	0.973170
recall	0.926582	0.984475	0.973252	0.955529	0.973252
f1-score	0.930706	0.983427	0.973252	0.957066	0.973206
support	790.000000	3285.000000	0.973252	4075.000000	4075.000000

Confusion Matrix:

```
[[ 732  58]
 [  51 3234]]
```

Test Result:

Accuracy Score: 96.98%

CLASSIFICATION REPORT:

	0	1	accuracy	macro avg	weighted avg
precision	0.933071	0.978281	0.969831	0.955676	0.969598
recall	0.908046	0.984517	0.969831	0.946282	0.969831
f1-score	0.920388	0.981389	0.969831	0.950889	0.969674
support	261.000000	1098.000000	0.969831	1359.000000	1359.000000

Confusion Matrix:

```
[[ 237  24]
 [  17 1081]]
```

### TestingResults:

```
new=X_test.iloc[]
a=np.asarray(ne)
a = a.reshape(1,1)
p=xgb_clf.predict()
if (p[0] == 1):
    print("PersonisaffectedbyCovid19andisatriskofdying")
else:
    print("Great!theresultsarenegativeandyoudon'thavetoworry")
```

- For i==5:  
Results:  
PersonisaffectedbyCovid19andisatriskofdying
- For i==12:  
Results:  
Great!theresultsarenegativeandyoudon'thavetoworry
- For i==50:  
Results:  
PersonisaffectedbyCovid19andisatriskofdying
- For i==16:  
Results:  
Great!theresultsarenegativeandyoudon'thavetoworry



## 7 CONCLUSION

In conclusion, the project on identifying individuals with COVID-19 using Python and Scikit-learn demonstrates the potential of machine learning in healthcare applications. By leveraging various classification algorithms, we were able to analyze patient data and identify patterns indicative of COVID-19 infection. The model's performance, assessed through metrics like accuracy, precision, and recall, highlighted the importance of data quality and feature selection in achieving reliable predictions.

The results of our analysis demonstrated a strong performance across multiple evaluation metrics, indicating the model's effectiveness. Key factors such as data quality, feature selection, and algorithm choice played a crucial role in optimizing the model's accuracy and reliability. This emphasizes the importance of comprehensive data preprocessing and thoughtful model selection in achieving meaningful outcomes in machine learning applications. Future work could involve expanding the dataset, incorporating additional features, and exploring more advanced algorithms to improve model robustness. Additionally, real-world implementation could pave the way for rapid screening and better resource allocation in healthcare settings. Overall, this project underscores the significant role that data science can play in addressing public health challenges.

## 8 REFERENCES

1. <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8049379/>
2. <https://www.sciencedirect.com/science/article/pii/S2667325822002862>
3. <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8138040/>
4. <https://www.geeksforgeeks.org/random-forest-regression-in-python/>
5. <https://www.mastersindatascience.org/learning/introduction-to-machine-learning-algorithms/decision-tree/>
6. <https://www.ibm.com/topics/logistic-regression>
7. <https://techieyantechnologies.com/2022/07/how-to-install-anaconda/>
8. <https://techieyantechnologies.com/2022/06/get-started-with-creating-new-environment-in-anaconda-configuring-jupyter-notebook-and-installing-libraries-using-requirements-txt-2/>

