DEPARTMENT OF COMPUTER SCIENCE & INFORMATION TECHNOLOGY

GURU GHASHIDAS UNIVERSITY BILASPUR CHHATTISGARH

A Major Project

ON

" SOIL FERTILITY PREDICTION "

Submitted in Partial Fulfilment of the Requirement for the Award of the Degree

of

MCA(NEW) : MASTER OF COMPUTER APPLICATION

Session 2022-2023

SUBMITTED BY

HARSHIT KUMAR SAHU

Roll No. – 21072123

Enrolment No. – GGV/21/05020

M.C.A 4th SEMESTER

UNDER THE GUIDENCE OF

Dr. AMIT KUMAR CHANDANAN

(Associate Professor)

DECLARATION OF THE CANDIDATE

We hereby certify that I am **Mr. HARSHIT KUMAR SAHU**, a student of M.C.A(Master of computer Application) Department of Computer science & Information Technology Session 2022-23 Enrolment no. GGV/21/05020 have carried out a project entitled "**SOIL FERTILITY PREDICTION**" under the Guidance of **Dr. AMIT KUMAR CHANDANAN** (Associate professor of GGU)

The matter presented in this report has not been submitted by me for the award of any other degree of this or any other Institute/University.

HARSHIT KUMAR SAHU

M.C.A 4th Semester

Dept of CSIT

GURU GHASIDAS UNIVERSITY BILASPPUR(C.G)

(A Central University)

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Date: **Dr. AMIT KUMAR CHANDANAN**

Place : Bilaspur **(Associate Professor)**

CERTIFICATE OF THE GUIDE

This is to certify that the project entitled **"Soil Fertility Prediction Model "** is a record of work carried out by **Mr. Harshit Kumar Sahu** under my guidance and supervision for the award of the Degree of MCA at Guru Ghasidas Vishwavidyalaya Bilaspur (C.G.). To the best of my knowledge and belief of the project, Embodies the work of the candidate himself, and has not been submitted for the award of any degree. Has duly been completed. Fulfils the requirement of the Ordinance relating to the MCA degree of the University. Is up to the desired standard in respect of contents and is being referred to the examiners.

(Signature of the guide)

Dr. AMIT KUMAR CHANDANAN

Associate Professor of

Guru Ghasidas Vishwavidyalaya,

Bilaspur, Chhattisgarh, India,

Recommendation of the Department

The Project work as mentioned above, here by being recommended and forwarded for examination and evaluation.

(Signature of the head of Department with seal)

Prof. A. K. Saxena

(H.O.D. of CSIT dept.)

GURU GHASHIDAS CENTRAL UNIVERSITY BILASPUR (C.G.)

DEPARTMENT OF COMPUTER SCIENCE & INFORMATION TECHNOLOGY

CERTIFICATE BY EXAMINER

This is to certified that the project work entitled **"Soil Fertility Prediction"** submitted by **Harshit Kumar Sahu** has completed under the guidance of **Dr. Amit Kumar Chandanan** (Associate Professor),of GGU bilaspur (c.g.) has been examined by the undersigned as a part of the examination for the award of the **MCA** (master of computer application) Degree in dept. Of "computer science & information technology" in G**uru Ghasidas Central University, Bilaspur (c.g.).**

Sign. _____

Internal Examiner

Date:

Sign. _____

External Examiner

Date:

ACKNOWLEDGEMENTS

It is indeed a great pleasure to express our sincere thanks to my supervisor **Dr. AMIT KUMAR CHANDANAN**, Associate Professor of Guru Ghasidas University, Bilaspur Chhattisgarh for his continuous support in this project. He was always there to listen and to give advice. He showed us different ways to approach a research problem and the need to be persistent to accomplish any goal. He taught us how to write academic paper, had confidence in us when we doubted ourselves, and brought out the good ideas in us. He was always there to meet and talk about our ideas, to proofread and mark up our paper, and to ask us good questions to help us think through our problems. Without his encouragement and constant guidance, we could not have finished this project.

Guru Ghasidas University, Bilaspur Chhattisgarh, **Mr. Amit saxena**, Head of Computer science & Information Technology Department really deserves our heartiest honour for providing us all the administrative support.

Last, but not least, we thank our parents, for giving us life in the first place, for educating us with aspects from both arts and sciences, for unconditional support and encouragement to pursue our interests. We dedicate this work to our parents who will feel very proud of us. They deserve real credit for getting us this far, and no words can ever repay for them.

Dr. AMIT KUMAR CHANDANAN

(Associate Professor)

ABSTRACT

The goal of this master's project is to provide a prediction model for the detection of Soil Fertility based on the chemical contents of soil .Soil fertility plays a crucial role in determining crop production volume. However, if the composition of soil nutrients, such as fertilizers, is not properly controlled and maintained, it can result in lower crop yields. Therefore, the measurement of soil nutrients is essential for achieving better plant growth and effective fertilization. Calcium (Ca), phosphorous (P), and pH level are among the key parameters commonly measured to monitor soil fertility as they provide important information for determining the soil's fertility status.

We used several important factors which effects soil fertility in our dataset and trained a model with various machine learning techniques like ANN, Decision tree, XG Boost, etc. To create best result and maximum proficiency in prediction.

SOIL FERTILITY PREDICTION

INDEX

INTRODUCTION

In the pursuit of sustainable agriculture and improved crop yields, effective soil management practices play a pivotal role. The delicate balance between crop production and maintaining soil health is a primary concern for modern farmers. This project focuses on harnessing the power of machine learning to revolutionize soil fertility management. By analysing soil data encompassing diverse chemical compositions, we aim to develop a sophisticated system that identifies optimal soil conditions and recommends targeted strategies to enhance fertility.

Soil management practices are crucial for boosting crop production while maintaining soil nutrients. Farmers need to determine the soil fertility requirements to achieve better and more cost-effective crop production. Soil pH is a critical soil parameter as it provides valuable information about various aspects of soil fertility. Major soil nutrients that contribute to yield production include phosphorus, potassium, nitrogen, calcium, and pH. Insufficient nutrient levels or excessive fertilization can result in lower crop yields. Therefore, it is essential to apply the appropriate quantity of fertilizer for optimal plant growth.

Data of different type of soils are collected with different chemical compositions and a data set is created which is then used in machine learning to find the best of the chemical composition for a fertile soil and to make the soil further fertile the machine learning will help to understand the current soil condition and soil quality of fertility can be increased with right kind of fertilizers.

BRIEF OVERVIEW OF THE TECHNOLOGY

1. **Python :** Python is a high-level, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation,It supports multiple programming paradigms, including structured, object-oriented and functional programming.

2. **Machine Learning :** a type of artificial intelligence (AI) that allows software applications to become more accurate at predicting outcomes without being explicitly programmed to do so. Machine learning algorithms use historical data as input to predict new output values.It Contain Various Tools and Libraries which we used in this project.

    - **Pandas :** an open source Python package that is most widely used for data science/data analysis and machine learning tasks.it helps in file handling in python like dataset for ML model.
    - **Numpy :** a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. Moreover Numpy forms the foundation of the Machine Learning stack.
    - **Scikit-learn :** Scikit-learn (Sk-learn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modelling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python.

HARDWARE & SOFTWARE REQUIREMENT

PROCESSOR TYPE	Pentium IV or above for optimum performance.
SYSTEM RAM	2.00GB and Above
INPUT DEVICE	BASIC KEYBOARD AND MOUSE
OUTPUT DEVICE	STANDARD COLOR MONITOR
OPERATING SYSTEM	WINDOWS 7,8,10
SOFTWARE	PYTHON AND TENSORFLOW OR PYTORCH

FEASIBILITY STUDY

Feasibility study is the measure of how beneficial or practical the development of an information
system will be to an organization. The Feasibility analysis is a cross life cycle activity and should
be continuously performed throughout the system life cycle.
Feasibility tests;

Operational feasibility:

By making application data dependent the trained model will help the user to easily operate it by putting the data in model.
Users will get a very quick service by Putting model in cloud. Also developers will feel comfortable by reduction of their work. Recording errors will be reduced. Easy to handle a large demand. Losing of data will be avoided.
Considering all these factors we can conclude that all the users and end users will be satisfied by the system.

Technical feasibility:

For the design and development of the system, several software products have been accommodated.
· Coding – Python or any other ML supported Language.
This software's have the enough efficiency in producing the system. Therefore the project is technically feasible.

Schedule feasibility:

The duration of time required for the project has been planned appropriately and it is the same as the duration of time expected by the customer. Therefore the product can be delivered to the customer within the expected time duration, satisfying the customer. Hence the project is feasible in scheduling.

Economic feasibility:

According to the resources available and the project scheduling process it is estimated that the expenses allocated for the software to be developed, by the customer is sufficient enough. Hence the economical factor has been considered feasible.

PROBLEMS ARE SOLVED BY SYSTEM

Here our main goal is that we answer the question of user in the particular manner with which the user gets satisfied and able to have the desired content. There are the some problems which may be solving by our application. Which are as follows:-

_ **Fast service: -** In the sense of fast service by this we can give the fast service to the user of the Soil Fertility Prediction Model.

_ **Reduces the workload: -** in the sense of workload reduction that means all the tasks are done by the automated system/machine which give the rest to the Farmers and correct knowledge about their soil fertility.

_ **Accuracy Prediction: -** in the sense of Accurate prediction our system provide the Accuracy in a Healthy manner and provide the good Architecture in all the operations of the model.

_ **Provide the satisfaction to the Users: -** In the sense of satisfaction of the user, because user is doing its all operations with the system on his requirements and he is completely satisfied because our system is fulfils its all requirements.

_ **Able to do all type of Soil Fertility Prediction: -** we are developing the type system which is able to do all type of Soil Prediction.

_ **Remove the complexity in the operations: -** our system is providing the real life features to providing the service since the user of the system in not feel the complexity in its operations.

_ **Networking:-**It covers the maximum space through its network for its service at any time and any where.

CHARACTERISTICS OF THE ENTITIES ARE USED TO SOLVE THE PROBLEM

Here we are use the big one entity which is our computer system. Which is use to run our project in the network environment. So we are use all the characteristics of the computer system to solve the problem all the problems which are occurs in Application some characteristics which are as follows:-

Fast accessibility: - by any computer system we can got the fast access for our operations. Which are take the more time in the manually form.

Multitasking: - by this character we can perform the more than on task at single time.

Storage space and Management: - by this character we have the large space to have all the records which are completely safe and sound in compare to the manually management.

Portability: - by this we can remove the problem of state of the customer that means customer can access all the operations from any place on its facility.

Mobility: - by this we can remove the problem of specific place and time. Which is always needed in manually system? Because our project is able to survive in the network environment.

_ **User Friendly Environment: –** Graphical user interface is used for easy handling of the system. A user can use the system efficiently without meeting any difficulty.

_ **User Satisfaction: -** Under consideration according user requirement and expectation, the system is developed.

_ **Security: -** System is secure in itself by checking the illegal access of invalid users. System is also capable to input some security checks at certain points

_ **Quick Response: -** As the processing time of any task is minimized therefore the user gets the quick response of his/her queries even though there is a huge amount of data.

_ **Robust Error Handling: -** The errors and undesirable situation, generated through the user intervention, are handled successfully to ensure that the system operates without halting by providing the appropriate error messages to the user.

_**Accuracy: -** All the processes provided by the system are accurate thus increase the reliability.

GENERAL CHARACTERISTICS

Introduction:

This section introduces the software and includes the characteristics and the constraints effecting
the product and requirements.

Product Perspective:

Product Function:

The Soil Fertility Prediction Model the will perform following functions:

_ User Input of Data
_ Select Run Button
_ Request for answer form model
_ Generate Answer
_ Display Answer

User Characteristics:

_ The users should have a basic computer literacy to work with a computer.
_ The users of this web application should know English to understand the given the relevant user documentation, reference materials and instructions.
_ The user should be able to follow simple instructions given by the system.

General Constraints:

The following are the general design constraints _ Data encoding scheme: ASCII character.
_ Total available memory for programming, logic, tables etc as specified in this document
should not be exceeded.

Assumptions and Dependencies:

_ The python Tensorflow or pytorch environment is accessible in real time
_ The flask server will be used to ascend to web page .

FUTURE SCOPE

The future of this project depend on whether the author has enough spare time over the next 2 month to continue with the developing. The author feels that last few remaining features would round off the system. If the author was to try to sell this system then more system testing would have to be done, in a particular a more comprehensive real – world. Testing environment would have to be adopted along with some real words usage. Multiple concurrent users would be command in real world usage but have been difficult to test for considering there was only tester involved in this project. This type of system would benefit for the hardware in case of a system failure for the software in case of newly found bugs, in return of a subscription free.

The proposed soil fertility prediction model lays the foundation for a multitude of promising future research directions. Advanced feature engineering techniques could be explored to extract more meaningful insights from soil data, potentially enhancing prediction accuracy. The integration of spatial and temporal data could lead to a more comprehensive understanding of soil dynamics. Additionally, the model's application could extend beyond prediction to offer real-time recommendations for soil management practices, optimizing resource allocation and promoting sustainable agriculture. Collaborative efforts with domain experts could further refine the model's performance and contribute to the development of a robust decision support system for farmers and agricultural stakeholders.

DATA COLLECTION

Data collection is the initial and crucial step in building a successful soil fertility prediction model. High-quality, relevant data serves as the fuel that powers machine learning algorithms, enabling them to make accurate predictions. The key sources of data for soil fertility prediction models include:

1. Soil Samples: Collecting soil samples from various locations within a field provides valuable insights into the spatial distribution of nutrient levels. These samples are analysed for key parameters such as pH, organic matter content, nutrient concentration, and texture.
2. Climate Data: Climate variables such as temperature, precipitation, and humidity play a vital role in influencing soil fertility. Integrating historical and real-time climate data into the model enhances its predictive accuracy.
3. Crop and Yield Data: Information about the types of crops grown, yield levels, and cropping patterns can help establish correlations between soil fertility and crop performance.
4. Remote Sensing Data: Satellite and drone imagery provide a bird's-eye view of the field, allowing for the detection of subtle variations in vegetation health and soil moisture content.
5. Management Practices: Data on agronomic practices, such as crop rotation, tillage, and fertilizer application, contribute to a holistic understanding of soil fertility dynamics.

The collection of data available at the time for the project is from Kaggel where various elements available using which we ae are able to predict and build a soil fertility prediction model.

	N	P	K	pH	EC	OC	S	Zn	Fe	Cu	Mn	B	Output
0	138	8.6	560	7.46	0.62	0.70	5.90	0.24	0.31	0.77	8.71	0.11	0
1	213	7.5	338	7.62	0.75	1.06	25.40	0.30	0.86	1.54	2.89	2.29	0
2	163	9.6	718	7.59	0.51	1.11	14.30	0.30	0.86	1.57	2.70	2.03	0
3	157	6.8	475	7.64	0.58	0.94	26.00	0.34	0.54	1.53	2.65	1.82	0
4	270	9.9	444	7.63	0.40	0.86	11.80	0.25	0.76	1.69	2.43	2.26	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...
875	351	10.7	623	7.96	0.51	0.29	7.24	0.36	4.69	0.69	11.03	0.69	1
876	264	9.0	486	7.24	0.47	0.10	3.92	0.35	8.26	0.45	7.98	0.40	1
877	276	9.2	370	7.62	0.62	0.49	6.64	0.42	3.57	0.63	6.48	0.32	1
878	320	13.8	391	7.38	0.65	1.07	5.43	0.58	4.58	1.02	13.25	0.53	2
879	264	10.3	475	7.49	0.74	0.88	10.56	0.45	7.36	1.87	10.63	0.63	0

880 rows × 13 columns

(Collection of Data Used)

DATA MINNING

Data mining involves the exploration of large datasets to discover meaningful and actionable information that might otherwise remain hidden. In agriculture, data mining enables researchers and farmers to gain insights into soil behavior, crop growth patterns, and nutrient interactions, thus facilitating evidence-based decision-making for improved productivity and sustainability.

Utilizing Soil Fertility Data

Soil fertility data, encompassing a plethora of attributes such as nutrient concentrations, pH levels, organic matter content, and more, provides a rich source of information that can be harnessed through data mining. Here's how data mining techniques can be applied to soil fertility data:

Pattern Recognition: Data mining algorithms can identify recurring patterns within soil fertility data that might indicate specific trends or anomalies. For instance, patterns in nutrient concentrations across different seasons or crop rotations can reveal insights about nutrient cycling and management practices.

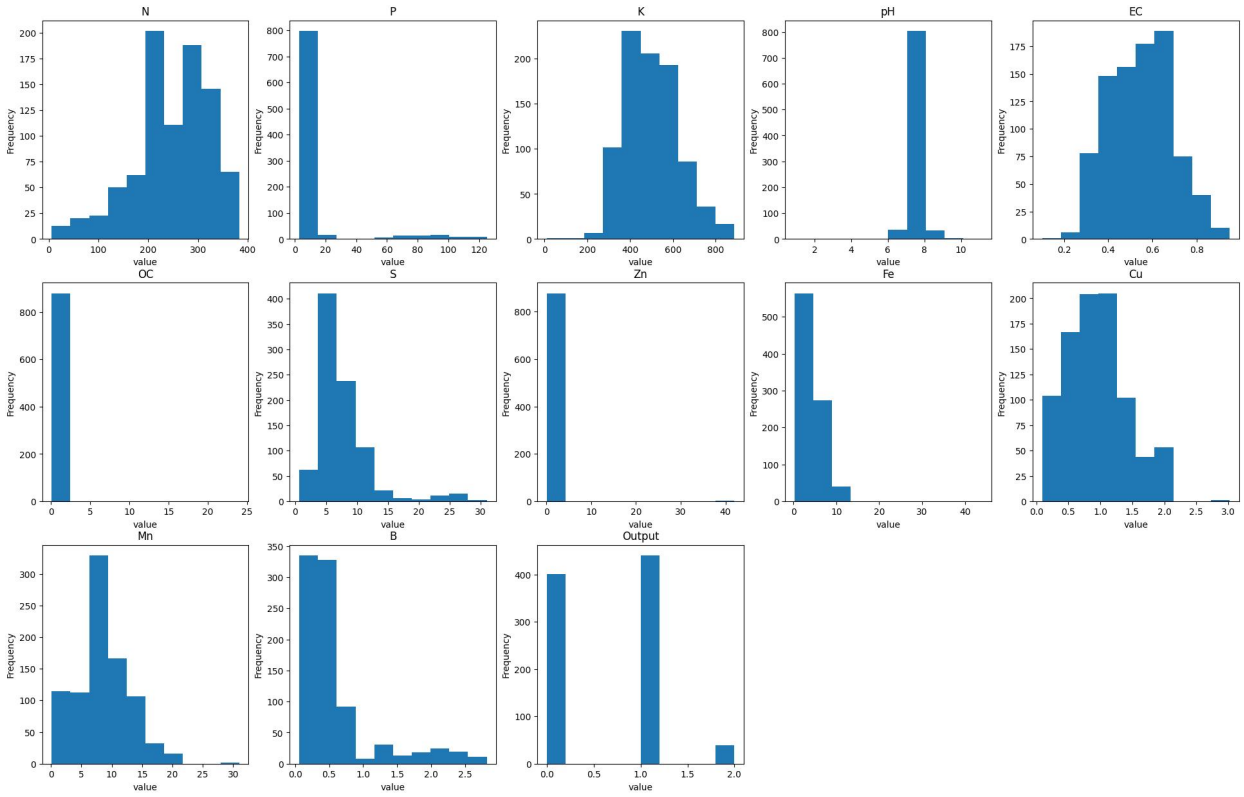
Clustering and Segmentation: Clustering algorithms group similar soil samples together, helping identify distinct soil types or regions with similar fertility characteristics. This knowledge can guide tailored management strategies for different areas of a field.

Association Rule Mining: This technique identifies relationships between different variables. It can help uncover associations between certain soil properties and crop yield, enabling farmers to prioritize specific nutrient adjustments for maximum output.

Predictive Modelling: Machine learning models, such as decision trees or neural networks, can be trained using historical soil fertility data to predict future soil nutrient levels. These models can inform nutrient application schedules and quantities.

Time Series Analysis: Data mining can reveal temporal trends in soil fertility data, offering insights into how nutrient levels change over time due to factors like climate fluctuations, irrigation practices, and cropping patterns.

Anomaly Detection: Data mining can identify outliers or anomalies in soil fertility data, indicating potential soil health issues that require immediate attention.



Data Visualization using *(matplotlib.pyplot)*

In above chart we can visualize the data and it's frequency based on which we can find pattern, as we see output has 3 clusters which are unbalanced than can be removed using some techniques like oversampling or under sampling .

DATA INTEGRITY

Ensuring data integrity is a critical aspect of building a reliable and accurate soil fertility prediction model. Data integrity involves maintaining the accuracy, consistency, and reliability of the data used for training and testing machine learning models. In the context of a soil fertility prediction model, data integrity is paramount to ensure that the model's predictions and recommendations are trustworthy and actionable. Here's how data integrity can be achieved for your soil fertility prediction model:

Data Collection Protocols:

1. Establish clear and standardized protocols for collecting soil samples. Ensure that the collection process is consistent across different sampling locations, times, and seasons.
2. Use proper equipment and techniques to prevent contamination of soil samples, which could lead to inaccurate nutrient measurements.

Data Quality Assurance:

1. Implement quality control measures during data collection, including regular calibration and maintenance of measurement instruments.
2. Cross-check and validate data against known standards or reference datasets to identify outliers or discrepancies.

Data Cleaning and Preprocessing:

1. Thoroughly clean the collected data by addressing missing values, outliers, and inconsistencies. Impute missing values using appropriate techniques, and carefully handle outliers that may be genuine or erroneous.
2. Standardize data formats and units to ensure consistency across different attributes.

Validation and Verification:

1. Split your dataset into training, validation, and testing subsets. Validation helps identify overfitting and ensures your model generalizes well to new data.
2. Perform independent verification of the model's predictions against new and unseen soil samples to assess its accuracy and generalizability.

Documentation:

1. Maintain detailed documentation of the data collection process, preprocessing steps, and any changes made to the dataset.
2. Document metadata such as sample locations, collection dates, and measurement techniques for each data point.

Data Security:

1. Implement appropriate security measures to protect sensitive soil fertility data from unauthorized access or breaches.
2. Utilize encryption and access controls to safeguard data during storage and transmission.

Expert Input:

1. Collaborate with domain experts, such as soil scientists or agronomists, to review and validate the data collection process and preprocessing methods.
2. Seek expert opinions when interpreting complex patterns or anomalies in the data.

Continuous Monitoring:

1. Regularly monitor the model's performance over time and recalibrate or update the model as needed to account for changing soil conditions or environmental factors.

Ethical Considerations:

1. Ensure that data collection and usage adhere to ethical standards and guidelines, respecting privacy, ownership, and legal regulations.

Transparency and Interpretability:

1. Choose machine learning algorithms that offer transparency and interpretability, making it easier to understand how the model arrives at its predictions.

Feedback Loop:

1. Establish a feedback loop with farmers and stakeholders to validate model predictions and gather real-world insights, which can help refine the model and improve its accuracy.

By diligently following these steps and prioritizing data integrity, your soil fertility prediction model can be built on a solid foundation, leading to more reliable and actionable insights for optimizing agricultural practices and enhancing crop productivity.

DATA CLEANING AND PREPROCESSING

Data cleaning and preprocessing are crucial steps in preparing your soil fertility dataset for building a reliable prediction model. These steps ensure that the data is accurate, consistent, and ready for analysis. Here's how you can achieve effective data cleaning and preprocessing for your soil fertility prediction model:

Handling Missing Values:

Identify and document missing values in your dataset.Depending on the extent of missing data, you can choose to impute missing values using techniques like mean, median, mode imputation, or more advanced methods like regression or k-nearest neighbours imputation. Be cautious when imputing data, as inappropriate imputation methods can introduce bias.

Dealing with Outliers:

Identify outliers in your data that may arise due to measurement errors or other anomalies. Decide whether to remove, transform, or cap outliers based on domain knowledge and the impact they may have on your model's performance. Visualization techniques like box plots and scatter plots can help identify outliers.

Data Transformation:

Normalize or standardize numerical features to bring them to a common scale. This helps algorithms converge faster and prevents certain features from dominating others. Log or power transformations can be applied to skewed data distributions to make them more symmetrical.

Encoding Categorical Variables:

Convert categorical variables (such as soil type, crop type, etc.) into numerical representations using techniques like one-hot encoding or label encoding. Ensure that the encoding method chosen is appropriate for the nature of the categorical data.

Handling Temporal Data:

If your dataset includes temporal information (e.g., time of sample collection), consider extracting relevant features like day of the week, month, or season to capture potential seasonality patterns.

Feature Engineering:

Create new features that encapsulate meaningful information from the existing ones. For example, you could calculate nutrient ratios or aggregate nutrient concentrations over different time periods.

Addressing Skewness:

Some machine learning algorithms perform better with normally distributed data. Apply transformations (e.g., logarithmic or square root) to address skewness in data distributions.

Removing Irrelevant Features:

Analyse the relevance of each feature in the context of soil fertility prediction. Remove features that are not expected to contribute significantly to the model's performance.

Data Consistency and Validation:

Ensure that data values fall within valid ranges based on domain knowledge. For instance, pH values should typically range between 0 and 14. Cross-validate data against external sources or reference datasets to validate consistency.

Splitting Data:

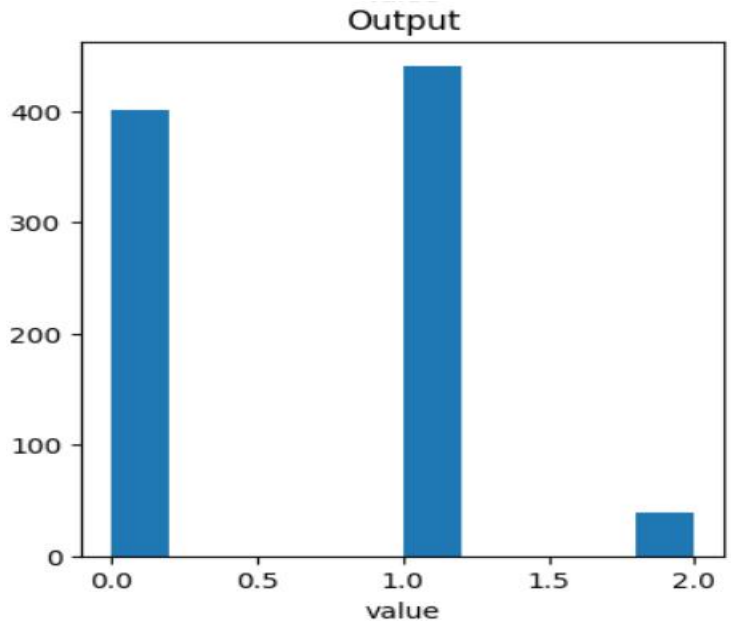
Divide your dataset into training, validation, and test sets to assess and optimize your model's performance. Consider using techniques like stratified sampling, especially if you have imbalanced classes or categorical variables.

The dataset was needed to be balanced as the output was unbalanced which could let to model over fitting problem so for solving that we performed the oversampling and under sampling techniques using which we balanced the dataset and completed the missing values

One way the imbalance may affect our Machine Learning algorithm is when our algorithm completely ignores the minority class. The reason this is an issue is because the minority class is often the class that we are most interested in.

	N	P	K	pH	EC	OC	S	Zn	Fe	Cu	Mn	B	Output
0	138	8.6	560	7.46	0.62	0.70	5.90	0.24	0.31	0.77	8.71	0.11	0
1	213	7.5	338	7.62	0.75	1.06	25.40	0.30	0.86	1.54	2.89	2.29	0
2	163	9.6	718	7.59	0.51	1.11	14.30	0.30	0.86	1.57	2.70	2.03	0
3	157	6.8	475	7.64	0.58	0.94	26.00	0.34	0.54	1.53	2.65	1.82	0
4	270	9.9	444	7.63	0.40	0.86	11.80	0.25	0.76	1.69	2.43	2.26	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...
875	351	10.7	623	7.96	0.51	0.29	7.24	0.36	4.69	0.69	11.03	0.69	1
876	264	9.0	486	7.24	0.47	0.10	3.92	0.35	8.26	0.45	7.98	0.40	1
877	276	9.2	370	7.62	0.62	0.49	6.64	0.42	3.57	0.63	6.48	0.32	1
878	320	13.8	391	7.38	0.65	1.07	5.43	0.58	4.58	1.02	13.25	0.53	2
879	264	10.3	475	7.49	0.74	0.88	10.56	0.45	7.36	1.87	10.63	0.63	0

880 rows × 13 columns



Original Data (Unbalanced)

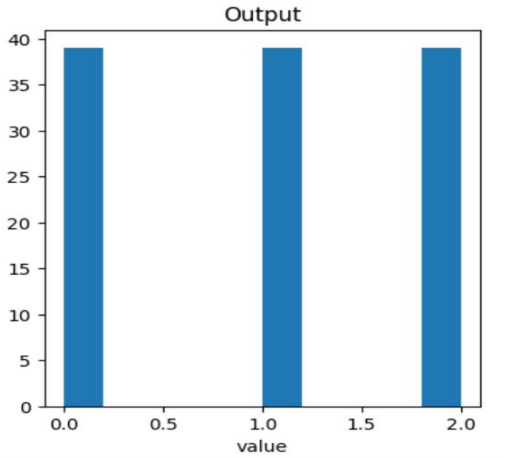
Under-sampling for Imbalanced Classification

Undersampling refers to a group of techniques designed to balance the class distribution for a classification dataset that has a skewed class distribution.

An imbalanced class distribution will have one or more classes with few examples (the minority classes) and one or more classes with many examples (the majority classes). It is best understood in the context of a binary (two-class) classification problem where class 0 is the majority class and class 1 is the minority class.

	N	P	K	pH	EC	OC	S	Zn	Fe	Cu	Mn	B	Output
0	220	5.5	391	7.23	0.29	0.29	4.22	0.33	6.32	0.96	12.05	0.34	0
1	88	9.0	539	7.14	0.29	1.07	7.84	0.42	6.32	0.86	14.00	0.53	0
2	138	8.1	718	7.48	0.63	0.94	12.60	0.19	1.02	1.15	2.24	2.47	0
3	251	6.8	528	7.20	0.24	0.68	9.65	0.29	9.32	0.53	11.32	0.29	0
4	201	4.8	496	7.50	0.53	0.49	5.43	0.25	5.69	2.02	7.68	0.65	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
112	358	12.9	422	7.02	0.48	1.27	5.43	0.29	3.56	0.63	11.56	0.42	2
113	345	13.6	444	7.85	0.38	0.10	6.33	0.35	8.54	1.02	9.64	0.65	2
114	314	14.9	359	7.04	0.43	0.68	5.13	0.17	6.11	0.88	3.74	0.21	2
115	333	13.8	401	7.62	0.51	0.49	3.92	0.26	3.05	1.03	9.05	0.34	2
116	320	13.8	391	7.38	0.65	1.07	5.43	0.58	4.58	1.02	13.25	0.53	2

117 rows × 13 columns



(Under Sampling Of Imbalanced Data)

The major drawback of random undersampling is that this method can discard potentially useful data that could be important for the induction process. The removal of data is a critical decision to be made, hence many the proposal of undersampling use heuristics in order to overcome the limitations of the non- heuristics decisions.That is why we are going to use Over Sampling because it also increases data in data set which could help in more accuracy models prediction.

SMOTE Oversampeling for Imbalanced Classification

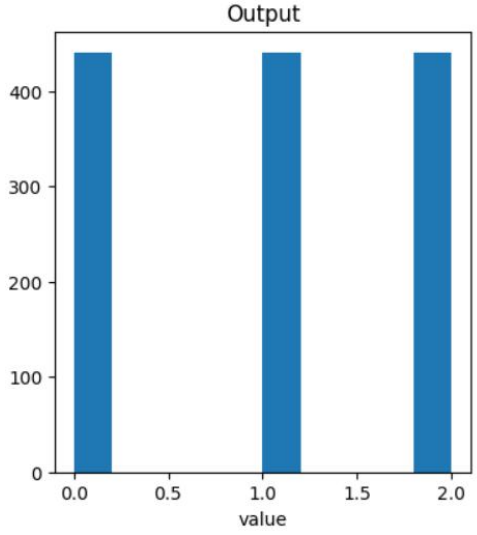
SMOTE is specifically designed to tackle imbalanced datasets by generating synthetic samples for the minority class. The SMOTE method is significant in dealing with class imbalance, focusing on its application in improving the performance of classifier models. By mitigating bias and capturing important features of the minority class, SMOTE contributes to more accurate predictions and better model performance.

SMOTE: Synthetic Minority Oversampling Technique

SMOTE is an oversampling technique where the synthetic samples are generated for the minority class. This algorithm helps to overcome the overfitting problem posed by random oversampling. It focuses on the feature space to generate new instances with the help of interpolation between the positive instances that lie together.

	N	P	K	pH	EC	OC	S	Zn	Fe	Cu	Mn	B	Output
0	138	8.600000	560	7.460000	0.620000	0.700000	5.900000	0.240000	0.310000	0.770000	8.710000	0.110000	0
1	213	7.500000	338	7.620000	0.750000	1.060000	25.400000	0.300000	0.860000	1.540000	2.890000	2.290000	0
2	163	9.600000	718	7.590000	0.510000	1.110000	14.300000	0.300000	0.860000	1.570000	2.700000	2.030000	0
3	157	6.800000	475	7.640000	0.580000	0.940000	26.000000	0.340000	0.540000	1.530000	2.650000	1.820000	0
4	270	9.900000	444	7.630000	0.400000	0.860000	11.800000	0.250000	0.760000	1.690000	2.430000	2.260000	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...
1315	361	12.257463	588	10.525000	0.534254	0.499328	9.786867	0.296791	3.523657	1.058358	9.182986	0.451418	2
1316	327	12.683795	437	7.595139	0.422808	0.405052	3.963414	0.535659	8.628016	0.676846	8.025433	0.561664	2
1317	321	18.908987	434	7.619319	0.416533	0.662106	3.972012	0.485201	6.789842	0.936781	7.373561	0.380681	2
1318	377	11.489057	632	7.800000	0.616256	0.548259	4.140674	0.370000	0.857288	0.300709	1.251817	0.897488	2
1319	321	14.361636	389	7.385616	0.641977	0.825441	10.652120	0.402837	6.802386	0.698140	8.433714	0.481605	2

1320 rows × 13 columns



(Result: SMOTE Applied on dataset)

ATTRIBUTE SELECTION

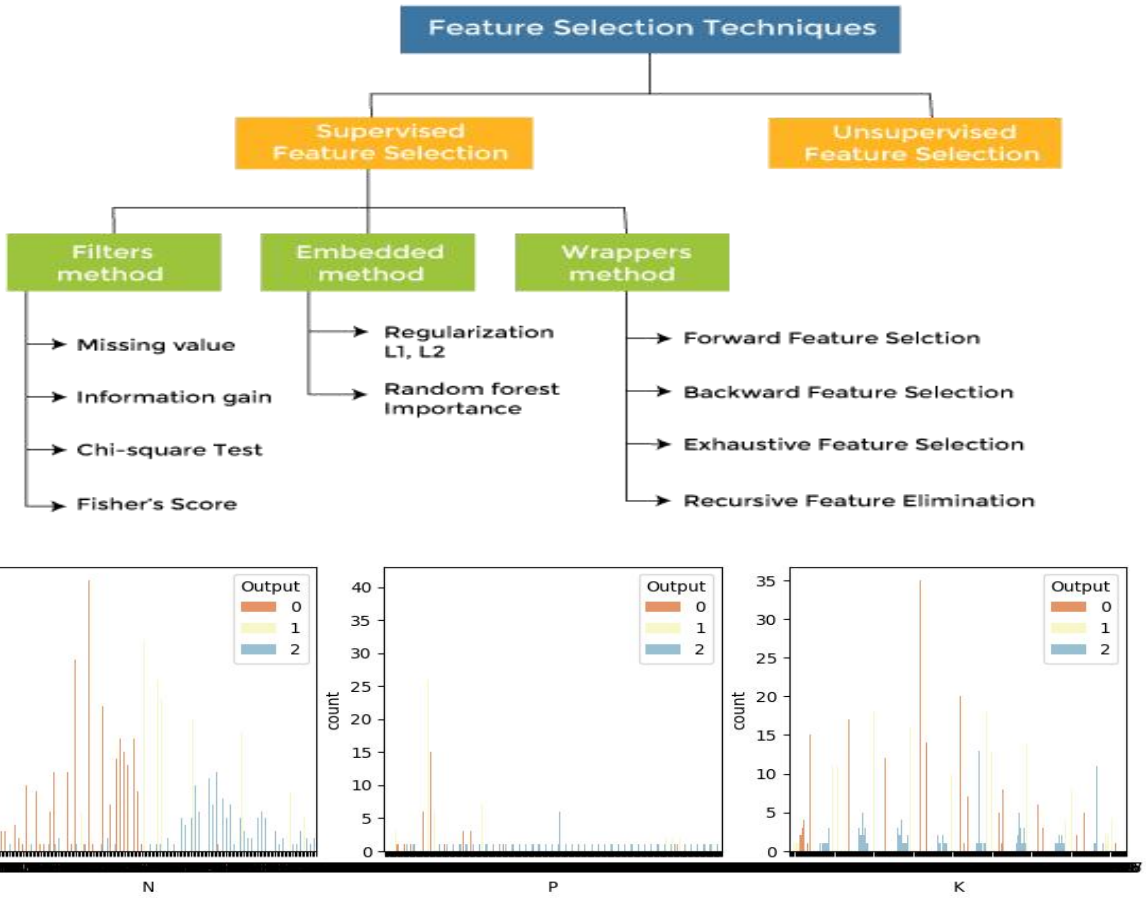
Attributes selection is the process of identifying and removing attributes from a training data set as much irrelevant and redundant attributes as possible. This reduces the dimensionality of the data. Many factors affect the success of machine learning on a given task. The representation and quality of the instance data is first and foremost. If there is much irrelevant and redundant information present or the data is noisy and unreliable, then knowledge discovery during the training phase is more difficult. In real world data, the representation of data often uses too many attributes, but only a few of them may be related to the target concept.

Generally attributes are characterized as

Relevant: These are attributes which have an influence on the output and their role cannot be assumed by the rest.

Irrelevant: Irrelevant attributes are defined as those attributes not having any influence on the output whose values are generated at random for each example.

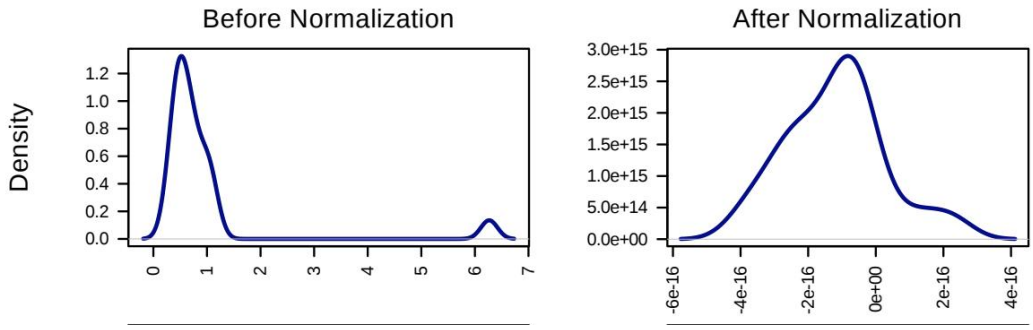
Redundant: A redundant exists, whenever a feature can take the role of another (perhaps the simplest way to model redundancy).



(Features Effecting the Output Most)

DATA NORMALIZATION

It is used whenever the attributes of the dataset have different ranges. It helps to enhance the performance and reliability of a machine learning model. Normalization is a scaling technique in Machine Learning applied during data preparation to change the values of numeric columns in the dataset to use a common scale. It is not necessary for all datasets in a model. It is required only when features of machine learning models have different ranges.

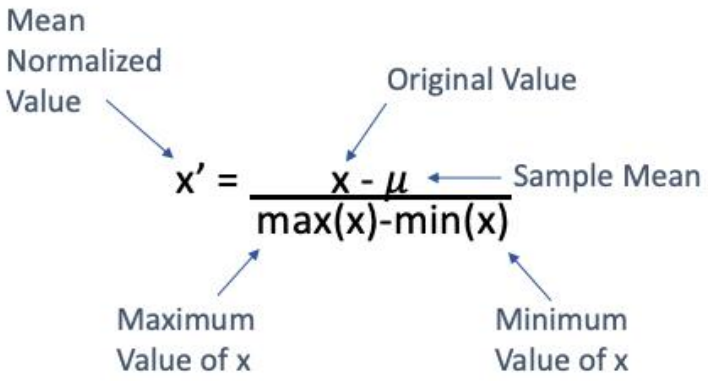


There are various normalization techniques you can use, such as Min-Max Scaling, Z-score (Standardization), and Robust Scaling. The choice of technique depends on the characteristics of your dataset and the requirements of your model.

a. Min-Max Scaling: Scales the features to a specific range, often between 0 and 1.

b. Z-score (Standardization): Transforms the features to have zero mean and unit variance.

c. Robust Scaling: Scales the features based on their median and interquartile range, making it more robust to outliers.



SPLITTING DATA

Splitting data is a fundamental step in machine learning where we divide the dataset into different subsets for the purpose of training, validation, and testing of machine learning model. This process helps you evaluate the performance of the model on unseen data and prevents over-fitting. The three main subsets typically used are:

Training Set:

The training set is the portion of your data that you use to train your machine learning model. During training, the model learns patterns, relationships, and features in the data. A larger training set often leads to a better-performing model, as it exposes the model to a diverse range of examples.

Validation Set:

The validation set is a separate subset of your data that you use to tune hyper-parameters, optimize the model, and prevent over-fitting. By monitoring the performance of your model on the validation set, you can make adjustments to improve its generalization ability. The validation set helps you choose the best configuration for your model before evaluating it on the test set.

Test Set:

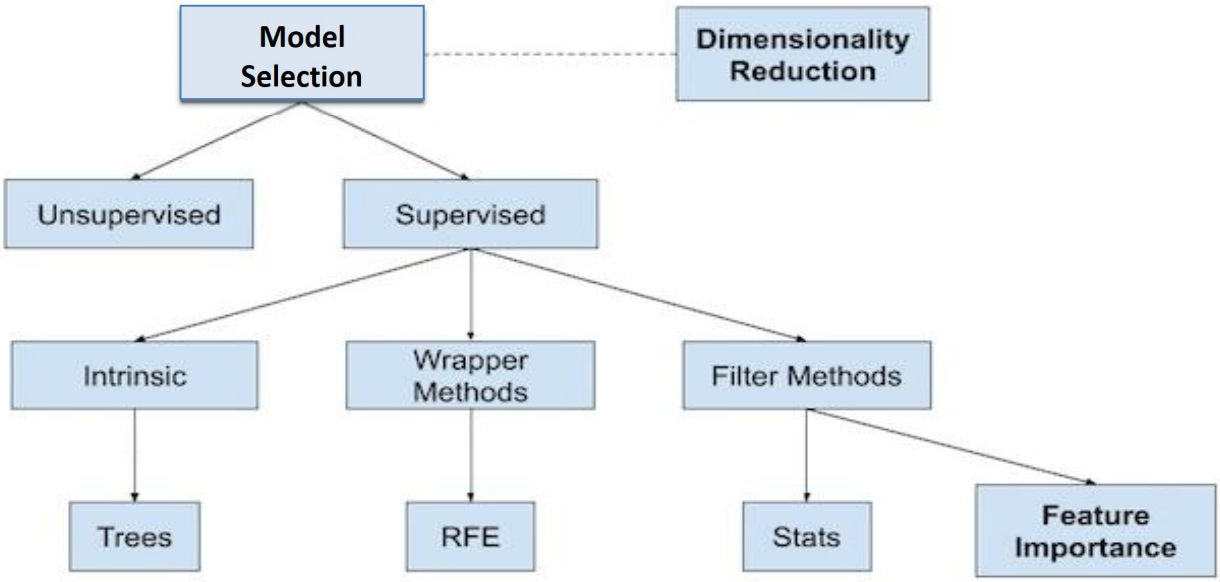
The test set is used to evaluate the final performance of your trained model. It represents unseen data that the model has not been exposed to during training or validation. Evaluating your model on the test set gives you an estimate of its real-world performance and helps you assess how well it generalizes to new, unseen data.

The data set is divided into two parts named as x and y used a part x which have selected attributes for training of model and the other part y for testing the model the splitting is done in formal way by calling train_test_split from sklearn.model_selection.

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.25, random_state = 0)
```

Selecting Machine Learning Models for Prediction

Selecting Machine Learning Model is very important part in prediction model building there are several types of model defined for machine learning but we choose the specific mode on the basis of dataset we have we need to see if the dataset is for Supervised or Unsupervised machine learning .



There Are several methods / models we used in this prediction technique, we are mainly doing a supervised learning model training and to better understand we are doing classification of dataset on the basis of moderate good or Great type of fertile soil so it is basically a multi class Classification model we are going to need classification models which are coming under supervised machine learning.The Models we are Using are Mentioned Below.

. Linear Regression

. Random Forest

. K-Nearest Neighbours

. Decision Tree

. Artificial Neural Network

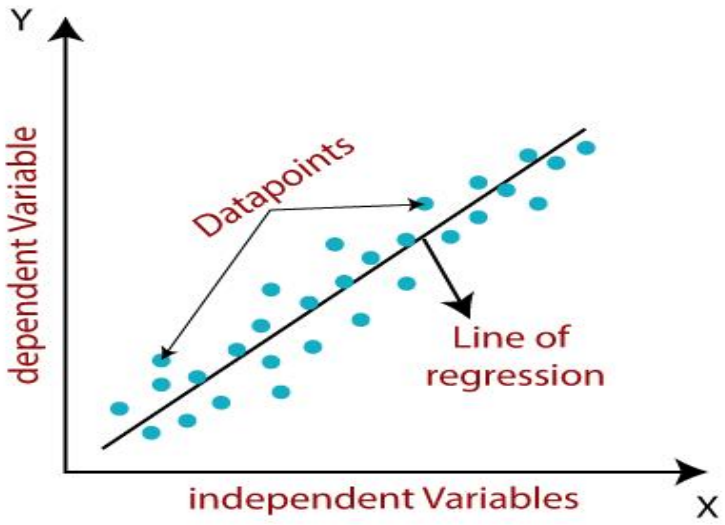
. SVM (Support Vector Machine)

. Gradient Boosting with XG Boost

Linear Regression

Linear regression is one of the easiest and most popular Machine Learning algorithms. It is a statistical method that is used for predictive analysis. Linear regression makes predictions for continuous/real or numeric variables such as **sales, salary, age, product price,** etc.

Linear regression algorithm shows a linear relationship between a dependent (y) and one or more independent (y) variables, hence called as linear regression. Since linear regression shows the linear relationship, which means it finds how the value of the dependent variable is changing according to the value of the independent variable.



When working with linear regression, our main goal is to find the best fit line that means the error between predicted values and actual values should be minimized. The best fit line will have the least error.

The different values for weights or the coefficient of lines (a_0, a_1) gives a different line of regression, so we need to calculate the best values for a_0 and a_1 to find the best fit line, so to calculate this we use cost function.

Using a Linear Regressor for soil fertility prediction can provide a straightforward and interpretable solution, especially when you suspect that there might be a linear relationship between the input features and the soil fertility levels. Linear regression is a simple yet powerful algorithm that is commonly used for regression tasks, where the goal is to predict a continuous numerical value based on input features.

Linear Regression can be useful for soil fertility prediction project:

Interpretability: Linear regression provides easily interpretable results. The coefficients associated with each feature in the model can help you understand the direction and magnitude

of their impact on soil fertility. For example, if the coefficient of a certain nutrient is positive, it indicates that an increase in that nutrient level is associated with an increase in soil fertility.

Simplicity: Linear regression is a straightforward algorithm that is relatively easy to implement and understand. It doesn't involve complex hyperparameters or intricate settings, making it a good starting point for modeling your soil prediction problem.

Baseline Model: Linear regression can serve as a baseline model. You can use its performance as a benchmark to compare against more complex algorithms. If a linear model performs well, it might suggest that the relationships in your data are indeed primarily linear.

Feature Importance: Linear regression can help identify the most important features for predicting soil fertility. The magnitude of the coefficients indicates the strength of the relationship between each feature and the target variable. This information can guide further feature engineering or selection.

Assumption Testing: Linear regression has assumptions, such as linearity, independence of errors, and homoscedasticity (constant variance). By applying diagnostic tests, you can assess whether these assumptions are met, which can provide insights into the quality and reliability of your model.

Low Complexity: Linear regression is computationally efficient and requires relatively low computational resources compared to more complex models like neural networks or ensemble methods. This can be advantageous when working with larger datasets.

However, it's important to note that linear regression also has limitations:

Limited to Linear Relationships: Linear regression assumes a linear relationship between the input features and the target variable. If your data exhibits complex, nonlinear patterns, linear regression may not capture them effectively.

Sensitivity to Outliers: Linear regression can be sensitive to outliers in the data, which might lead to skewed predictions. Preprocessing and outlier handling techniques are important to mitigate this issue.

Underfitting: If the relationships between features and soil fertility are not linear, a linear regressor may underperform more flexible models like Random Forest or Gradient Boosting.

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression(random_state = 0)
model.fit(X_train, y_train)
```

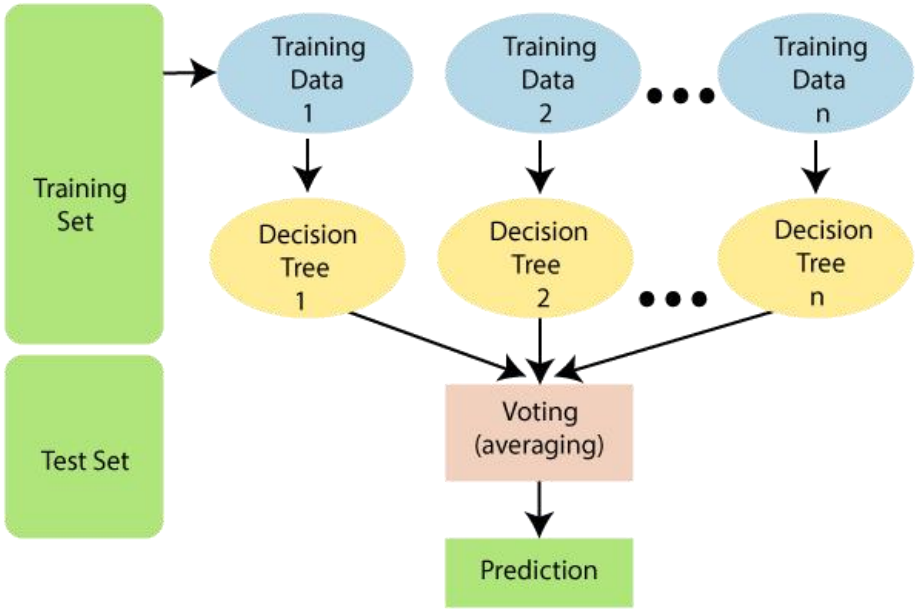
(Implementation of Logistic Regression)

Random Forest

Random Forest is an ensemble technique capable of performing both regression and classification tasks with the use of multiple decision trees and a technique called Bootstrap and Aggregation, commonly known as bagging. The basic idea behind this is to combine multiple decision trees in determining the final output rather than relying on individual decision trees.

Random Forest has multiple decision trees as base learning models. We randomly perform row sampling and feature sampling from the dataset forming sample datasets for every model. This part is called Bootstrap.

We need to approach the Random Forest regression technique like any other machine learning technique.



As the name suggests, "Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset." Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.

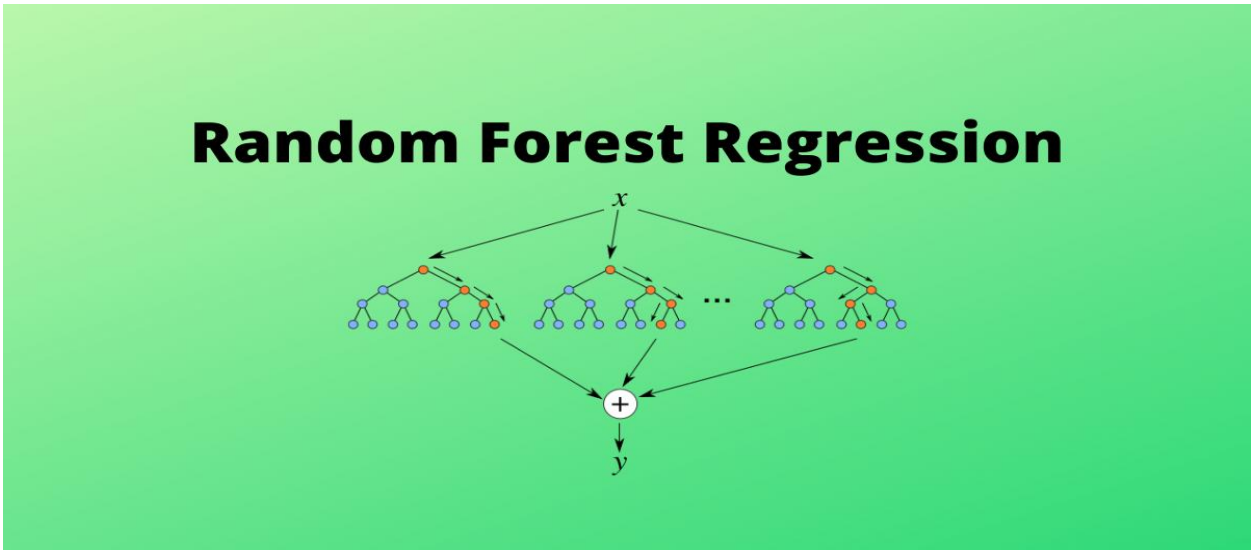
A Random Forest is an ensemble learning technique that combines multiple decision trees to create a robust and accurate prediction model. In the context of regression, it builds a forest of decision trees and aggregates their predictions to produce a final continuous value (in your case, soil fertility levels).

Key Features and Benefits for Your Project:

Handling Non-linearity: Soil fertility is influenced by a multitude of factors, many of which may interact in non-linear ways. Random Forest can capture these intricate relationships, making it suitable for predicting soil fertility that might exhibit complex behavior.

Feature Importance: Random Forests can provide insight into feature importance. This means you can identify which soil properties and characteristics are most influential in determining

fertility. This information can help agricultural experts and researchers understand the key factors affecting soil health.



Robustness to Outliers: Outliers and noisy data are common in real-world datasets. Random Forests are less sensitive to outliers compared to single decision trees, which can lead to more stable and accurate predictions.

Ensemble Learning: By combining multiple decision trees, a Random Forest reduces the risk of overfitting, which is crucial when working with limited or noisy data. The ensemble nature of Random Forest helps generalize well to unseen data.

Reducing Bias and Variance: The random selection of subsets of data and features during tree construction helps to reduce both bias and variance, leading to better model performance.

Easy to Tune: Although Random Forests have hyperparameters that need tuning (e.g., number of trees, maximum depth), they are relatively easier to work with compared to some other complex models like deep neural networks.

Interpretability: While Random Forests are not as interpretable as individual decision trees, they can still provide insight into feature importance and relationships. This can help in explaining your model's predictions to stakeholders.

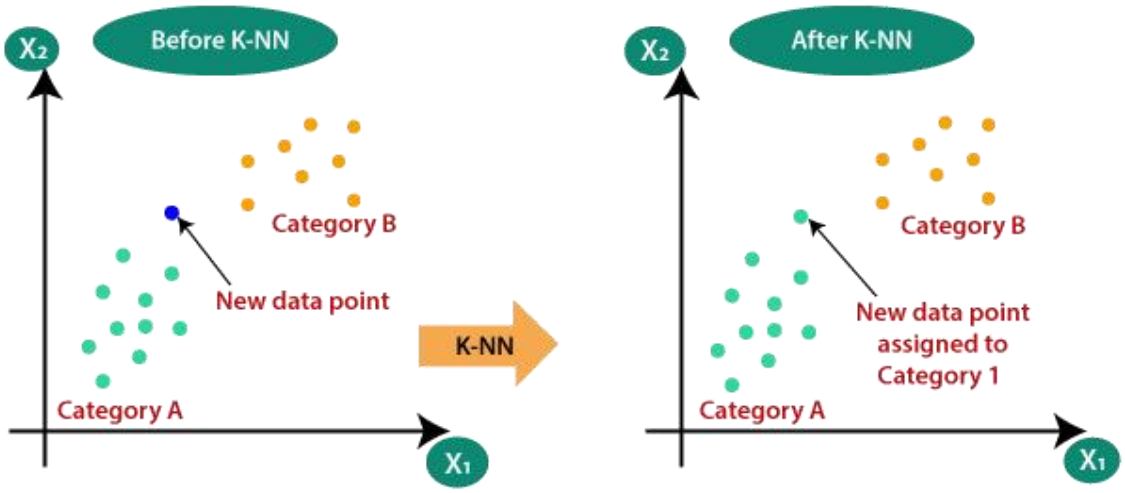
```
# Fitting Random Forest Regression to the dataset
# import the regressor
from sklearn.ensemble import RandomForestRegressor

# create regressor object
regressor = RandomForestRegressor(n_estimators=100, random_state=0)

# fit the regressor with x and y data
regressor.fit(X, Y)
```

(Implementation of Random Forest Regressor)

K-Nearest Neighbours

K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique. K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories. K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.

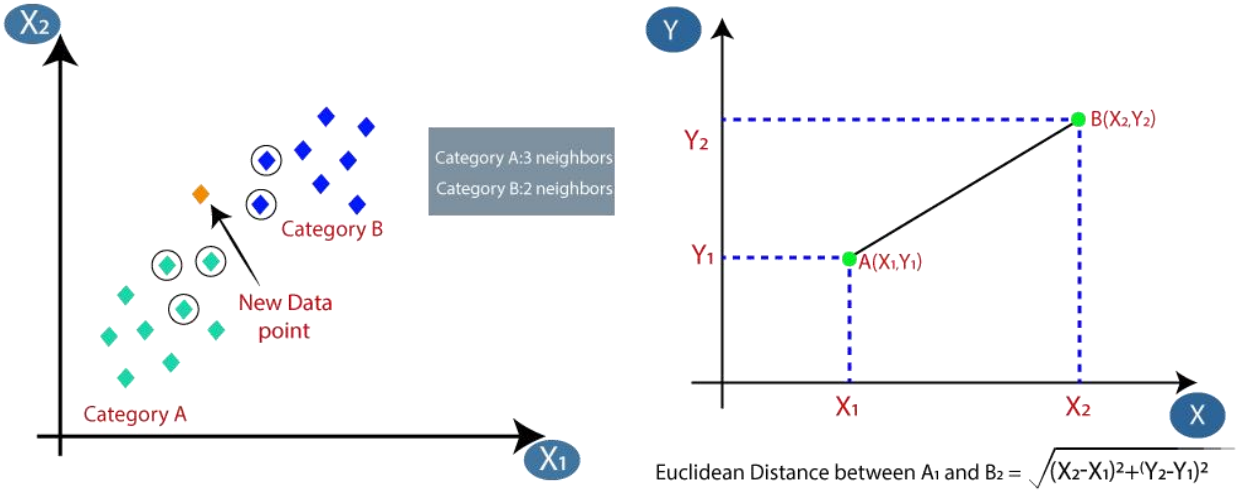
K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems. K-NN is a non-parametric algorithm, which means it does not make any assumption on underlying data. It is also called a lazy learner algorithm because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset. KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.



Suppose there are two categories, i.e., Category A and Category B, and we have a new data point x1, so this data point will lie in which of these categories. To solve this type of problem, we need a K-NN algorithm. With the help of K-NN, we can easily identify the category or class of a particular dataset.

Below are some points to remember while selecting the value of K in the K-NN algorithm:

- There is no particular way to determine the best value for "K", so we need to try some values to find the best out of them. The most preferred value for K is 5.
- A very low value for K such as K=1 or K=2, can be noisy and lead to the effects of outliers in the model.
- Large values for K are good, but it may find some difficulties.



Advantages of KNN Algorithm:

- It is simple to implement.
- It is robust to the noisy training data
- It can be more effective if the training data is large.

Disadvantages of KNN Algorithm:

- Always needs to determine the value of K which may be complex some time.
- The computation cost is high because of calculating the distance between the data points for all the training samples.

Now we will fit the K-NN classifier to the training data. To do this we will import the KNeighborsClassifier class of Sklearn Neighbors library. After importing the class, we will create the Classifier object of the class. The Parameter of this class will be n_neighbors: To define the required neighbors of the algorithm. Usually, it takes 7. metric='minkowski': This is the default parameter and it decides the distance between the points. p=2: It is equivalent to the standard Euclidean metric.

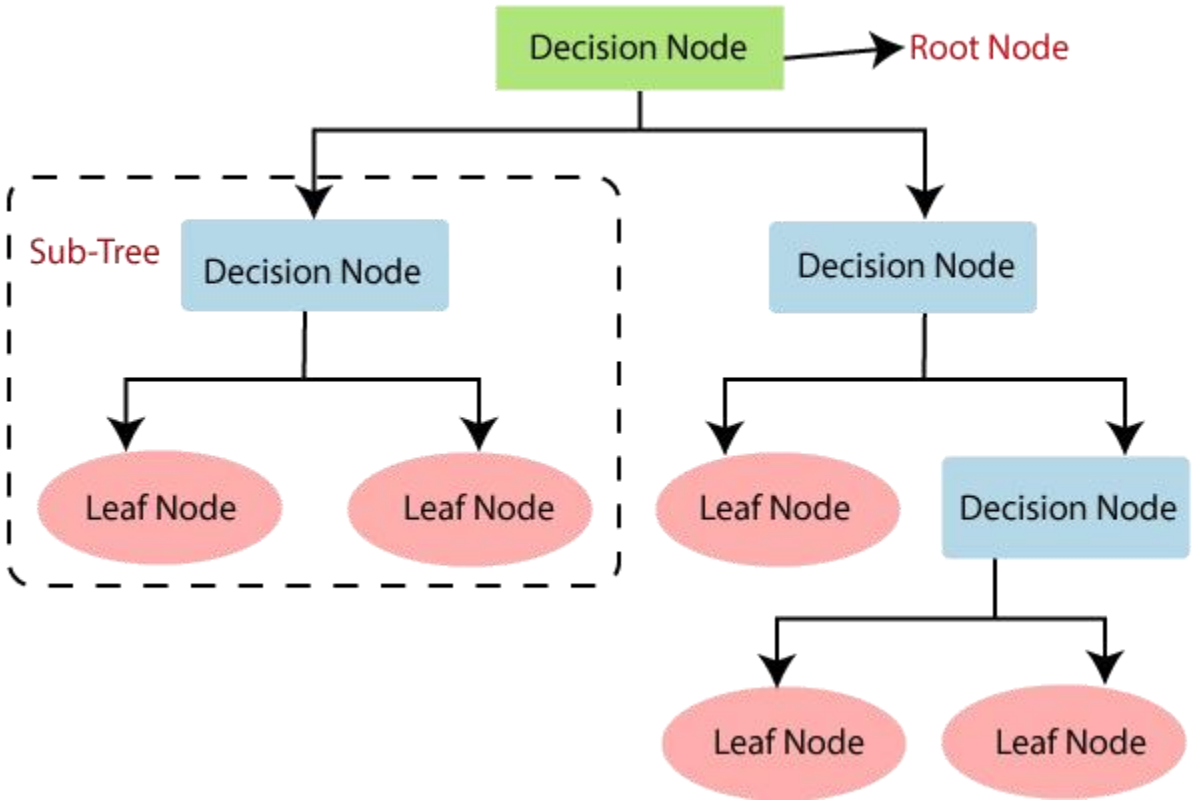
```
from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier(n_neighbors=7)

knn.fit(X_train, y_train)
```

(Implementation of K-Nearest Neighbours)

Decision Tree

A decision tree is one of the most powerful tools of supervised learning algorithms used for both classification and regression tasks. It builds a flowchart-like tree structure where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (terminal node) holds a class label. It is constructed by recursively splitting the training data into subsets based on the values of the attributes until a stopping criterion is met, such as the maximum depth of the tree or the minimum number of samples required to split a node.

During training, the Decision Tree algorithm selects the best attribute to split the data based on a metric such as entropy or Gini impurity, which measures the level of impurity or randomness in the subsets. The goal is to find the attribute that maximizes the information gain or the reduction in impurity after the split.



Decision Tree 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.

In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node. Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches. The decisions or the test are performed on the basis of features of the given dataset. *It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions.* It is called a decision tree because, similar to a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure. In order to build a tree, we use the CART algorithm, which stands for Classification and Regression Tree algorithm.

A decision tree simply asks a question, and based on the answer (Yes/No), it further split the tree into sub-trees.

- **Root Node:** Root node is from where the decision tree starts. It represents the entire dataset, which further gets divided into two or more homogeneous sets.
- **Leaf Node:** Leaf nodes are the final output node, and the tree cannot be segregated further after getting a leaf node.
- **Splitting:** Splitting is the process of dividing the decision node/root node into sub-nodes according to the given conditions.
- **Branch/Sub Tree:** A tree formed by splitting the tree.
- **Pruning:** Pruning is the process of removing the unwanted branches from the tree.
- **Parent/Child node:** The root node of the tree is called the parent node, and other nodes are called the child nodes.

Advantages of the Decision Tree

- It is simple to understand as it follows the same process which a human follow while making any decision in real-life.
- It can be very useful for solving decision-related problems.
- It helps to think about all the possible outcomes for a problem.
- There is less requirement of data cleaning compared to other algorithms.

Disadvantages of the Decision Tree

- The decision tree contains lots of layers, which makes it complex.
- It may have an over-fitting issue, which can be resolved using the **Random Forest algorithm.**
- For more class labels, the computational complexity of the decision tree may increase.

```
from sklearn.tree import DecisionTreeClassifier

# Creating the classifier object
clf_gini = DecisionTreeClassifier(criterion = "gini",random_state = 100,max_depth=3, min_samples_leaf=5)

# Performing training
clf_gini.fit(X_train, y_train)

# Decision tree with entropy
clf_entropy = DecisionTreeClassifier(
            criterion = "entropy", random_state = 100,
            max_depth = 3, min_samples_leaf = 5)

# Performing training
clf_entropy.fit(X_train, y_train)
```

(Implementation of Decision Tree)

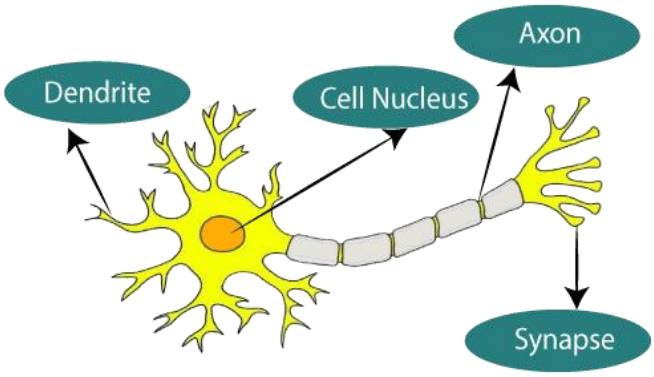
Artificial Neural Network

Artificial Neural Network Tutorial provides basic and advanced concepts of ANNs. Our Artificial Neural Network tutorial is developed for beginners as well as professions.

The term "Artificial neural network" refers to a biologically inspired sub-field of artificial intelligence modelled after the brain. An Artificial neural network is usually a computational network based on biological neural networks that construct the structure of the human brain. Similar to a human brain has neurons interconnected to each other, artificial neural networks also have neurons that are linked to each other in various layers of the networks. These neurons are known as nodes.

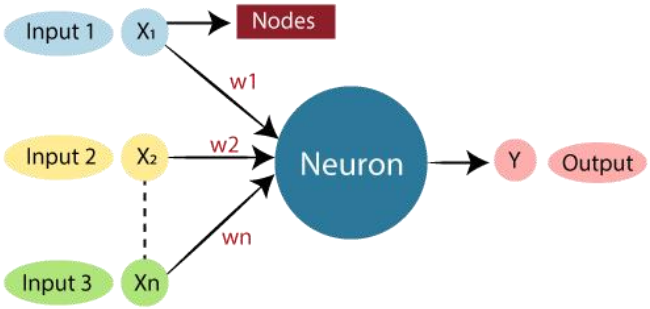
Artificial neural network tutorial covers all the aspects related to the artificial neural network. In this tutorial, we will discuss ANNs, Adaptive resonance theory, Kohonen self-organizing map, Building blocks, unsupervised learning, Genetic algorithm, etc.

The term "Artificial Neural Network" is derived from Biological neural networks that develop the structure of a human brain. Similar to the human brain that has neurons interconnected to one another, artificial neural networks also have neurons that are interconnected to one another in various layers of the networks. These neurons are known as nodes.



(The given figure illustrates the typical diagram of Biological Neural Network.)

Dendrites from Biological Neural Network represent inputs in Artificial Neural Networks, cell nucleus represents Nodes, synapse represents Weights, and Axon represents Output.



(The typical Artificial Neural Network looks something like the given above figure)

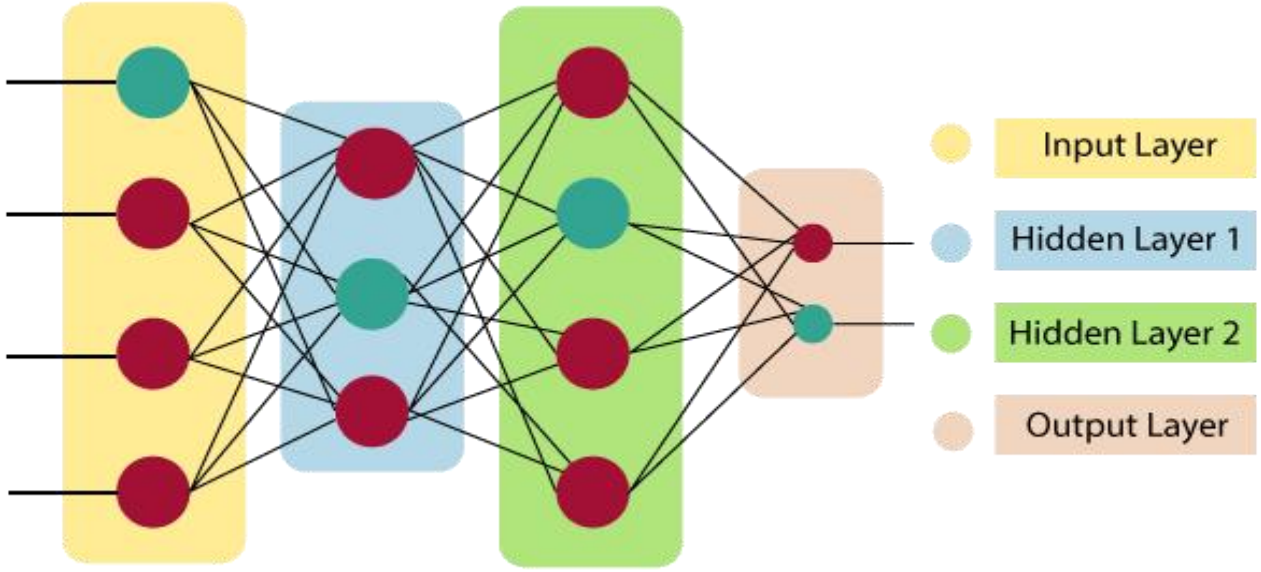
Relationship between Biological neural network and artificial neural network:

Biological Neural Network	Artificial Neural Network
Dendrites	Inputs
Cell nucleus	Nodes
Synapse	Weights
Axon	Output

The architecture of an artificial neural network:

To understand the concept of the architecture of an artificial neural network, we have to understand what a neural network consists of. In order to define a neural network that consists of a large number of artificial neurons, which are termed units arranged in a sequence of layers. Lets us look at various types of layers available in an artificial neural network.

Artificial Neural Network primarily consists of three layers:



Input Layer: As the name suggests, it accepts inputs in several different formats provided by the programmer.

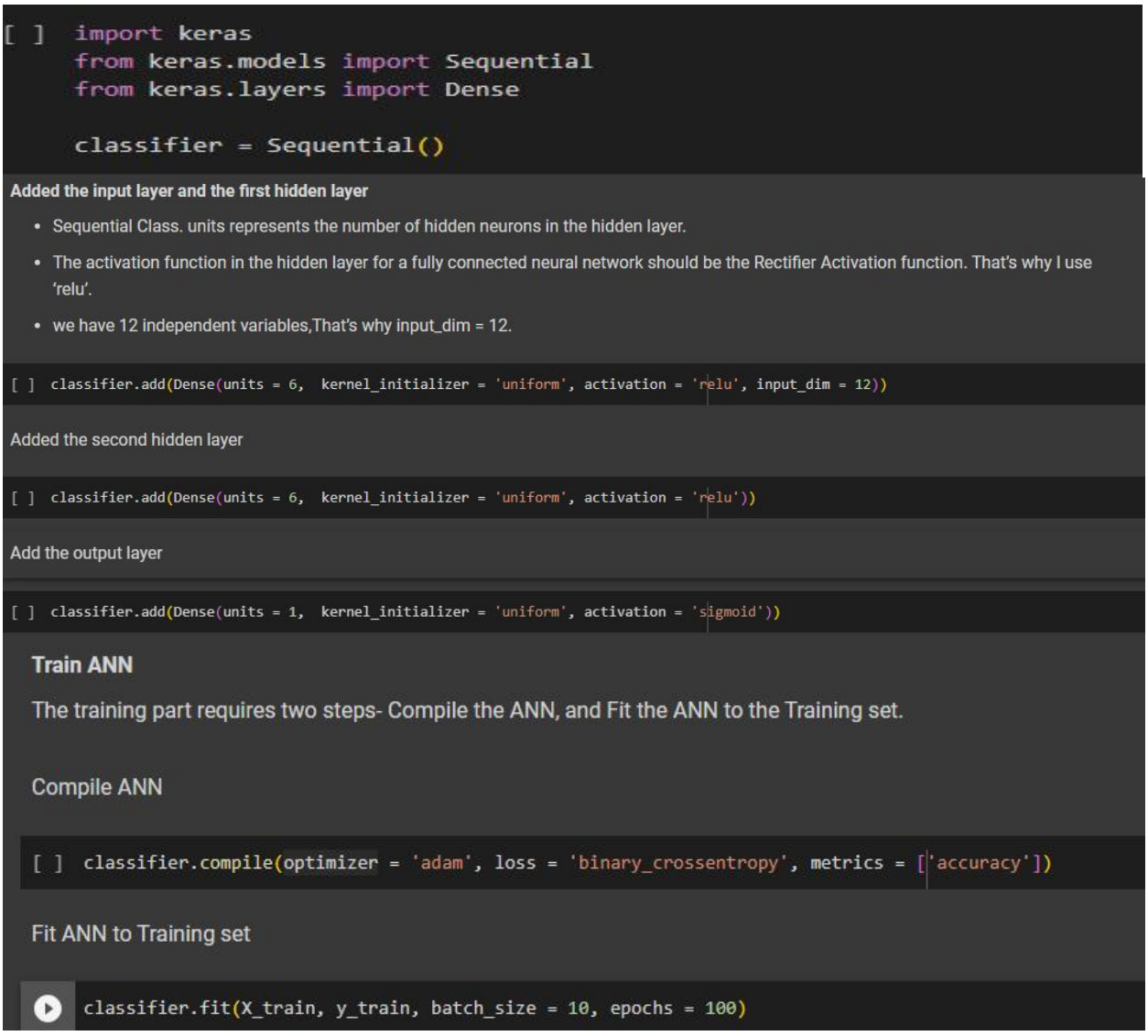
Hidden Layer: The hidden layer presents in-between input and output layers. It performs all the calculations to find hidden features and patterns.

Output Layer: The input goes through a series of transformations using the hidden layer, which finally results in output that is conveyed using this layer.

The artificial neural network takes input and computes the weighted sum of the inputs and includes a bias. This computation is represented in the form of a transfer function.

$$\sum_{i=1}^{n} Wi * Xi + b$$

It determines weighted total is passed as an input to an activation function to produce the output. Activation functions choose whether a node should fire or not. Only those who are fired make it to the output layer. There are distinctive activation functions available that can be applied upon the sort of task we are performing.



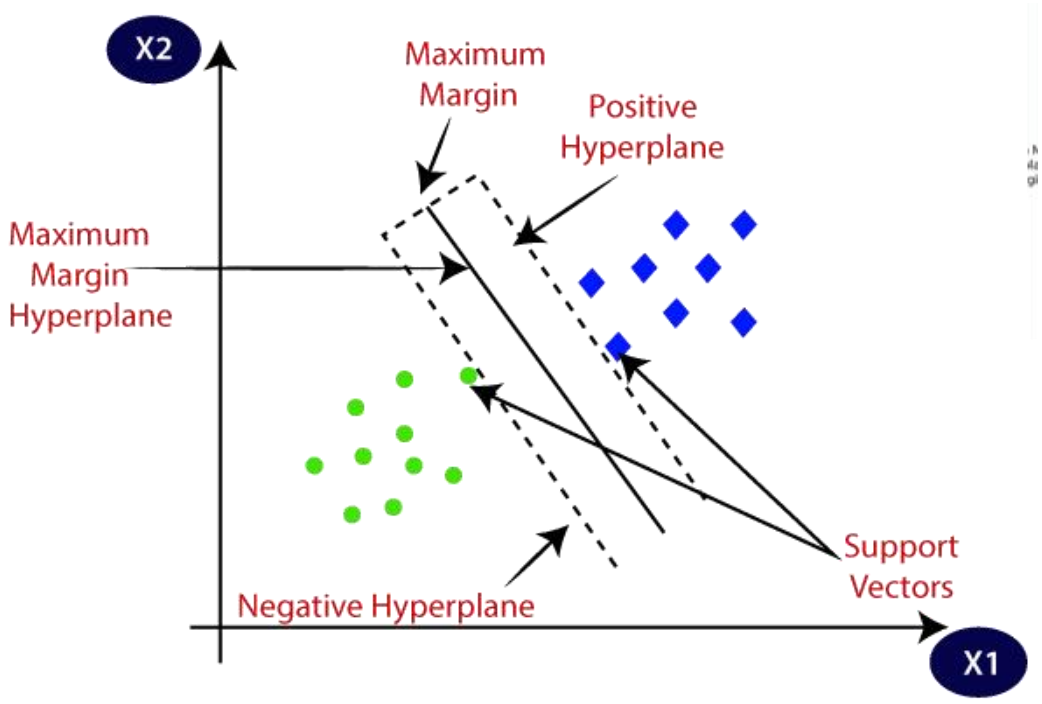
(Implementation of Artificial Neural Network)

SVM (SUPPORT VECTOR MACHINE)

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.

The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine. Consider the below diagram in which there are two different categories that are classified using a decision boundary or hyperplane:

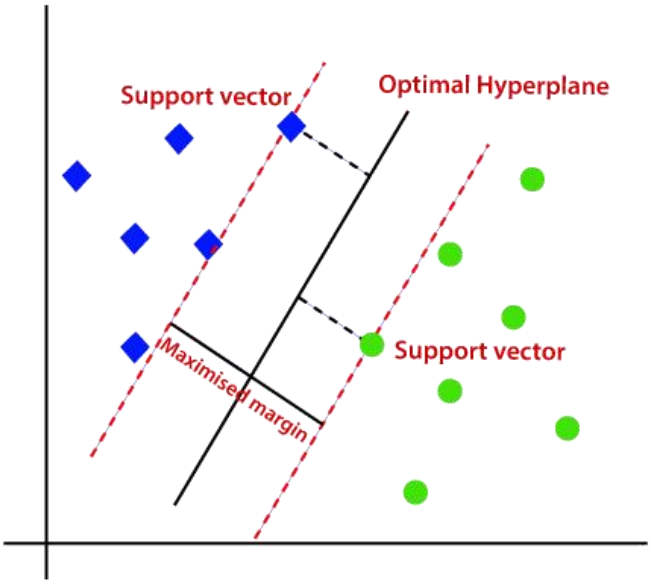


SVM can be of two types:

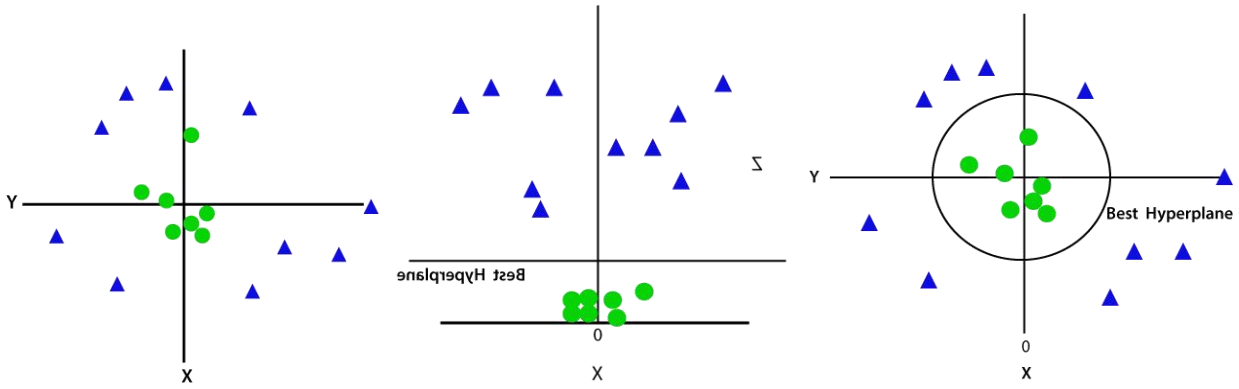
- **Linear SVM:** Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier.
- **Non-linear SVM:** Non-Linear SVM is used for non-linearly separated data, which means if a dataset cannot be classified by using a straight line, then such data is termed as non-linear data and classifier used is called as Non-linear SVM classifier.

Hence, the SVM algorithm helps to find the best line or decision boundary; this best boundary or region is called as a **hyperplane**. SVM algorithm finds the closest point of the lines from both the classes. These points are called support vectors. The distance between the vectors and the

hyperplane is called as **margin**. And the goal of SVM is to maximize this margin. The **hyperplane** with maximum margin is called the **optimal hyperplane**.



If data is linearly arranged, then we can separate it by using a straight line, but for non-linear data, we cannot draw a single straight line



(Non-Linera Support Vector Machine)

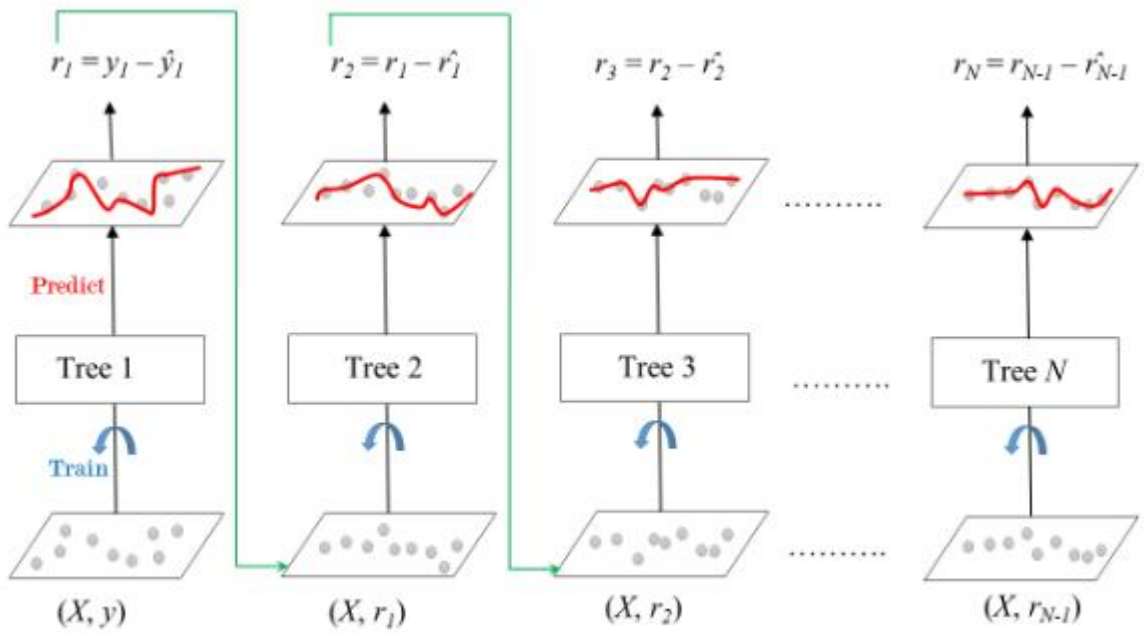
```
from sklearn.svm import SVC
svc = SVC()
svc.fit(X_train, y_train)
```

(Implementation of Support Vector Machine)

Gradient Boosting with XG Boost

Gradient Boosting is a popular boosting algorithm in machine learning used for classification and regression tasks. Boosting is one kind of ensemble Learning method which trains the model sequentially and each new model tries to correct the previous model. It combines several weak learners into strong learners.

Gradient Boosting is a powerful boosting algorithm that combines several weak learners into strong learners, in which each new model is trained to minimize the loss function such as mean squared error or cross-entropy of the previous model using gradient descent. In each iteration, the algorithm computes the gradient of the loss function with respect to the predictions of the current ensemble and then trains a new weak model to minimize this gradient. The predictions of the new model are then added to the ensemble, and the process is repeated until a stopping criterion is met.



XGBoost is an optimized Gradient Boosting Machine Learning library. It is originally written in C++, but has API in several other languages. The core XGBoost algorithm is parallelizable i.e. it does parallelization within a single tree.

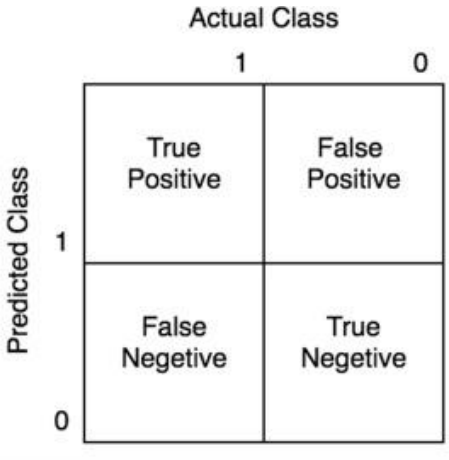
```
import xgboost as xgb

# Create the XGBoost classifier
xgb_model = xgb.XGBClassifier()

# Train the model
xgb_model.fit(X_train, y_train)
```

(Implementation of Gradient Boosting with XG Boost)

CONFUSION MATRIX

A confusion matrix is a matrix that summarizes the performance of a machine learning model on a set of test data. It is often used to measure the performance of classification models, which aim to predict a categorical label for each input instance. The matrix displays the number of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN) produced by the model on the test data.

For binary classification, the matrix will be of a 2X2 table, For multi-class classification, the matrix shape will be equal to the number of classes i.e for n classes it will be nXn. In machine Learning, Classification is the process of categorizing a given set of data into different categories. In Machine Learning, To measure the performance of the classification model we use the confusion matrix.



- **True Negative:** Model has given prediction No, and the real or actual value was also No.
- **True Positive:** The model has predicted yes, and the actual value was also true.
- **False Negative:** The model has predicted no, but the actual value was Yes, it is also called as **Type-II error**.
- **False Positive:** The model has predicted Yes, but the actual value was No. It is also called a **Type-I error.**

It evaluates the performance of the classification models, when they make predictions on test data, and tells how good our classification model is. It not only tells the error made by the classifiers but also the type of errors such as it is either type-I or type-II error. With the help of the confusion matrix, we can calculate the different parameters for the model, such as accuracy, precision, etc.

n = total predictions	Actual: No	Actual: Yes
Predicted: No	True Negative	False Positive
Predicted: Yes	False Negative	True Positive

Classification Accuracy: It is one of the important parameters to determine the accuracy of the classification problems. It defines how often the model predicts the correct output. It can be calculated as the ratio of the number of correct predictions made by the classifier to all number of predictions made by the classifiers. The formula is given below:

$$\text{Accuracy}=\frac{TP+TN}{TP+FP+FN+TN}$$

Misclassification rate: It is also termed as Error rate, and it defines how often the model gives the wrong predictions. The value of error rate can be calculated as the number of incorrect predictions to all number of the predictions made by the classifier. The formula is given below:

$$\text{Error rate}=\frac{FP+FN}{TP+FP+FN+TN}$$

Precision: It can be defined as the number of correct outputs provided by the model or out of all positive classes that have predicted correctly by the model, how many of them were actually true. It can be calculated using the below formula:

$$\text{Precision}=\frac{TP}{TP+FP}$$

Recall: It is defined as the out of total positive classes, how our model predicted correctly. The recall must be as high as possible.

$$\text{Recall}=\frac{TP}{TP+FN}$$

F-**measure:** If two models have low precision and high recall or vice versa, it is difficult to compare these models. So, for this purpose, we can use F-score. This score helps us to evaluate the recall and precision at the same time. The F-score is maximum if the recall is equal to the precision. It can be calculated using the below formula:

$$\text{F-measure}=\frac{2*Recall*Precision}{Recall+Precision}$$

COMPARING ACCURACY

ML MODELS	ACCURACY % NORMAL DATA	ACCURACY % UNDERSAMPELING	ACCURACY % OVERSAMPELING (SMOTE)
Logistic Regression	87.72%	63.33%	72.12%
Random Forest	45.90%	36.66%	30.30%
K-Neatest Neighbors	80.45%	63.33%	77.57%
Decision Tree	90.45%	96.66%	88.78%
Artificial Neural Network	85.45%	30%	52.12%
Support Vector Machine	83.63%	63.33%	88.18%
XG Boost	89.09%	86.66%	94.54%

As we see the most accuracy is obtained by Decision Tree in undersampeling of dataset but it possibilities of under-fitting is keep occurring in undersampeling dataset so we need to consider the oversampled data where XG Boost model provides 94.54% highest in all of other models so can consider it .

CONCLUDING MODEL

Based on the accuracy results of different machine learning models under different data scenarios (normal, undersampling, and oversampling with SMOTE), here is the conclusion:

1. Normal Data:

- The Decision Tree model performs exceptionally well with an accuracy of 90.45%.
- Other models, such as Logistic Regression, K-Nearest Neighbors, Support Vector Machine, and XG Boost, also exhibit relatively high accuracies ranging from 83.63% to 89.09%.
- Random Forest and Artificial Neural Network show lower accuracy compared to other models.

2. Undersampling:

- The Decision Tree model maintains its high accuracy (96.66%).
- Logistic Regression, K-Nearest Neighbors, and Support Vector Machine see reduced accuracy when compared to normal data.
- Random Forest, Artificial Neural Network, and XG Boost remain relatively less accurate.

3. Oversampling (SMOTE):

- **The XG Boost model achieves the highest accuracy of 94.54% under the oversampling scenario.**
- Decision Tree and Support Vector Machine also maintain relatively high accuracies.
- Logistic Regression, K-Nearest Neighbors, and Artificial Neural Network show mixed results, with varying accuracy values.
- Random Forest has the lowest accuracy among all models when using oversampling.

Recommendations:

Decision Tree: The Decision Tree model consistently performs well across all scenarios. It might be a reliable choice if New dataset is used , especially given its high accuracy and interpretability.

XG Boost: Under the oversampling scenario, XG Boost demonstrates the highest accuracy. This indicates that it could be a strong option when addressing class imbalance through oversampling techniques like SMOTE.

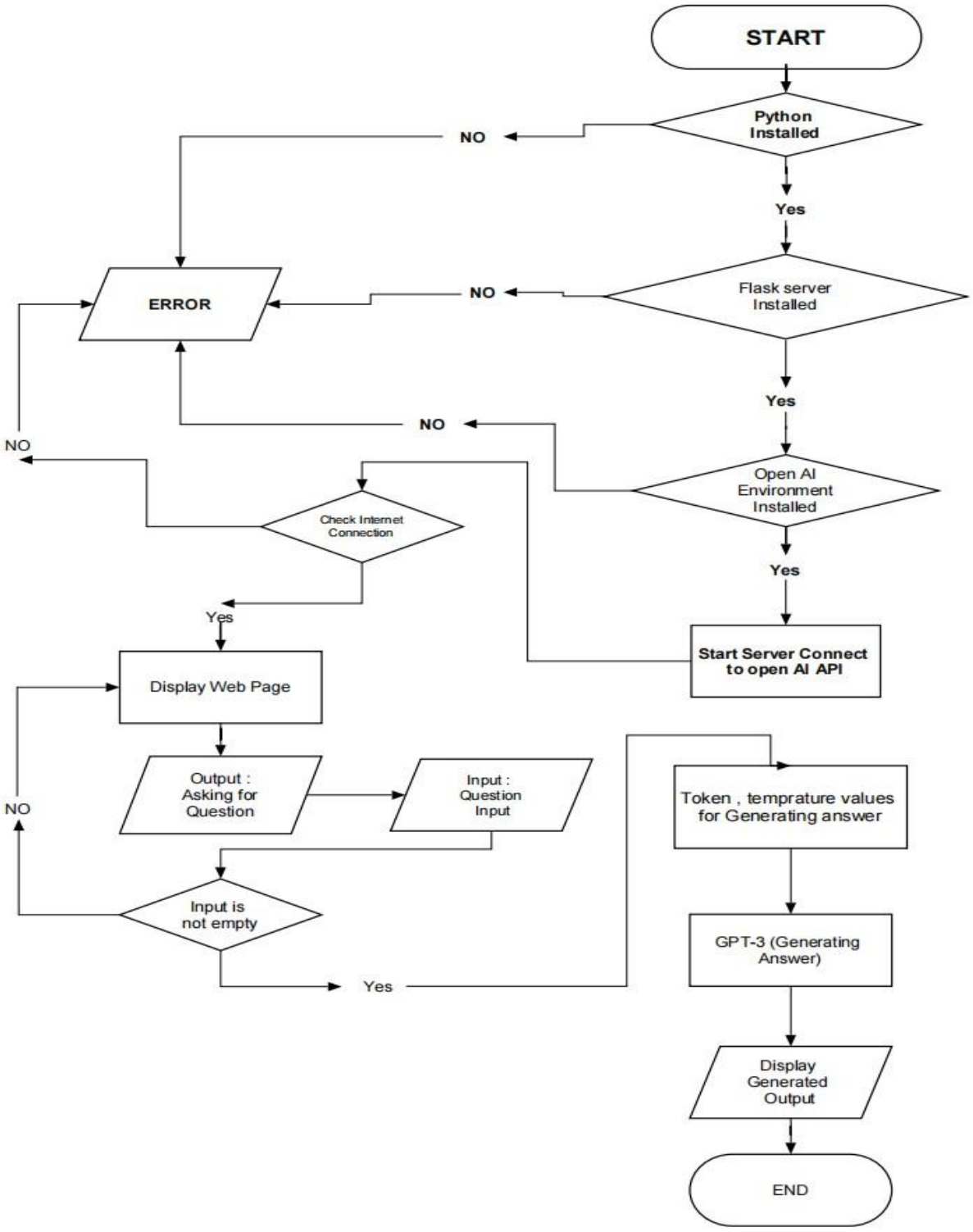
Support Vector Machine: SVM also maintains relatively high accuracy across scenarios, which makes it a competitive choice.

Logistic Regression: While Logistic Regression's accuracy decreases under undersampling and oversampling, it still performs reasonably well. It could be a good option if you value interpretability.

K-Nearest Neighbors: K-NN's accuracy is relatively stable, and it could be a suitable choice, especially considering its decent performance even with class imbalance.

Random Forest and Artificial Neural Network: These models seem to be more sensitive to the class imbalance introduced by undersampling and oversampling, leading to reduced accuracy. Depending on other considerations, you might explore other algorithms before these.

FLOW CHART


START
Python Installed
NO
Yes
Flask server Installed
NO
ERROR
Yes
Open AI Environment Installed
NO
NO
Check Internet Connection
Yes
Yes
Start Server Connect to open AI API
Display Web Page
Output : Asking for Question
Input : Question Input
NO
Token , temprature values for Generating answer
Input is not empty
GPT-3 (Generating Answer)
Yes
Display Generated Output
END

DATA FLOW DIAGRAM

A data flow diagram is a conceptual representation of the data flow that are required by Programmer. The first step in designing a DFD to develop an Robust program. The DFD serves as a blue print from which a Working flow of program maybe deduced. Figure 6.1.1 shows the DFD for the project.

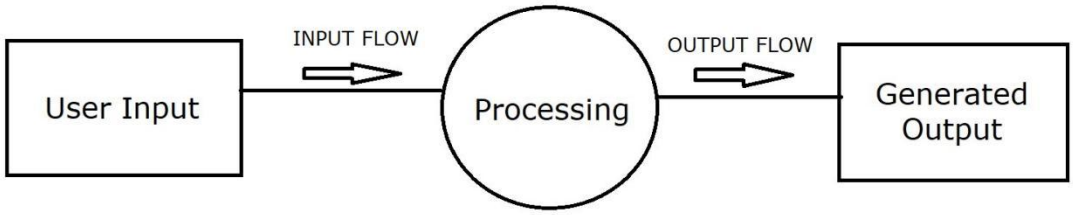


Figure6.1.1 Data Flow Diagram 0 LEVEL

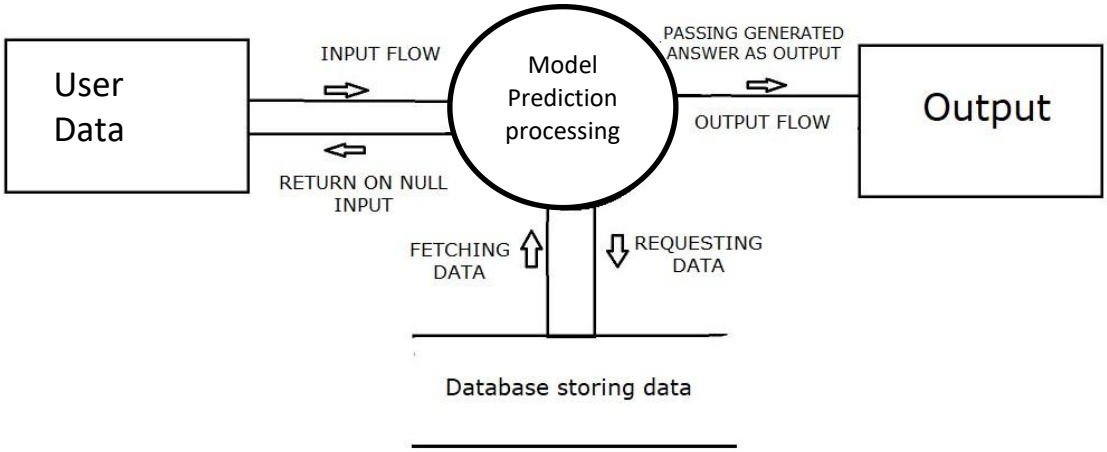


Figure6.1.1 Data Flow Diagram 1 LEVEL

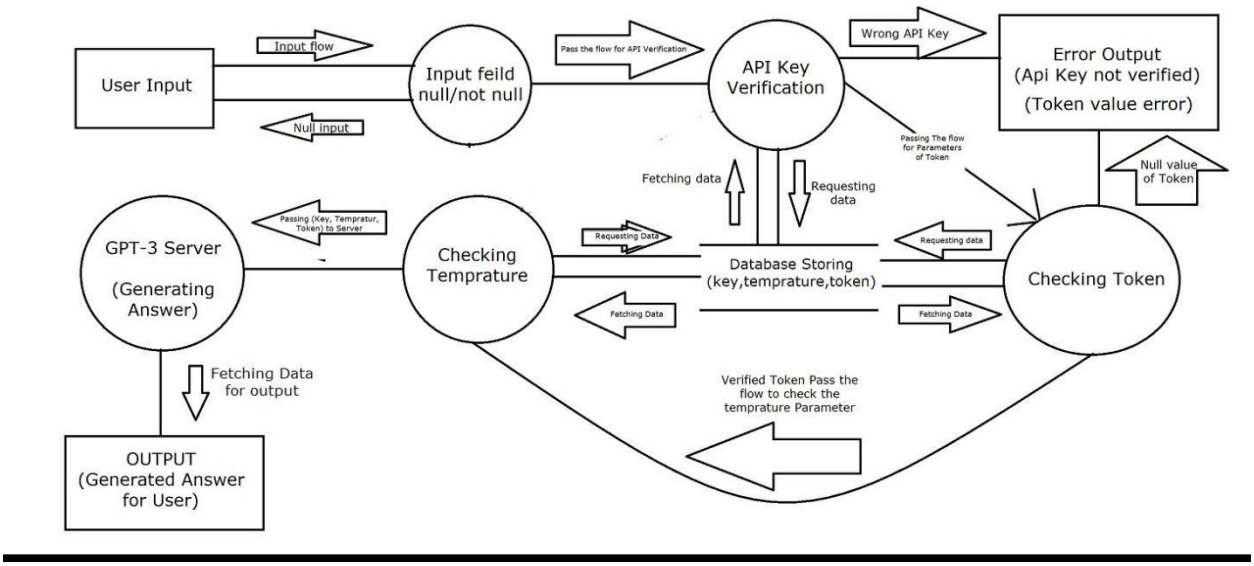


Figure6.1.1 Data Flow Diagram 2 LEVEL

METHEDOLOGY

The dataset used in this study encompasses diverse soil characteristics such as composition, pH, temperature, and moisture. Machine learning models including Logistic Regression, Random Forest, K-Nearest Neighbors, Decision Tree, Artificial Neural Network, Support Vector Machine, and XG Boost are employed for soil fertility prediction.The methodology section of prediction model study outlines the procedures and steps you followed to achieve your research goals. In the context of your soil fertility prediction study, the methodology involves how you processed the data, trained and evaluated the machine learning models, and tested them under different scenarios.

The models are evaluated in three data scenarios:

Normal Data: Using the original dataset.

Undersampling: Balancing the classes by reducing the dominant class instances.

Oversampling (SMOTE): Synthetic minority oversampling technique to balance class distribution.

RESULT

The results of our experiments demonstrate varying performances of the machine learning models across different data scenarios:

Normal Data:

Decision Tree excels with an accuracy of 90.45%, showcasing its ability to capture complex relationships in the data.

Logistic Regression, K-Nearest Neighbors, Support Vector Machine, and XG Boost also exhibit competitive accuracies ranging from 83.63% to 89.09%.

Undersampling:

Decision Tree maintains its high accuracy (96.66%), highlighting its robustness against class imbalance.

Logistic Regression, K-Nearest Neighbors, and Support Vector Machine experience reduced accuracy, indicating sensitivity to data manipulation.

Oversampling (SMOTE):

XG Boost achieves the highest accuracy of 94.54%, demonstrating its effectiveness in addressing class imbalance.

Decision Tree and Support Vector Machine maintain competitive performance.

Logistic Regression, K-Nearest Neighbors, and Artificial Neural Network display varying accuracy levels.

As we see the most accuracy is obtained by Decision Tree in undersampeling of dataset but it possibilities of under-fitting is keep occurring in undersampeling dataset so we need to consider the oversampled data where XG Boost model provides 94.54% highest in all of other models so can consider it .

CONCLUSION

The soil fertility prediction model developed in this study presents a promising avenue for agricultural decision-making. The Decision Tree model consistently stands out for its accuracy and interpretability. Additionally, XG Boost demonstrates superior performance under oversampling, showcasing its potential for managing class imbalance. The results underscore the importance of selecting the appropriate machine learning algorithm based on data characteristics and objectives. Further research could delve into ensemble methods and advanced feature engineering to enhance prediction accuracy and provide deeper insights into soil fertility determinants.

The vast amount of data that is now being collected alongside agricultural crops holds immense potential and must be thoroughly analysed to extract its full value. In my research, I have discovered that XG Boost, with its accuracy of 94.54%, is an excellent choice as a foundational algorithm for predicting soil fertility. By incorporating additional meta-algorithms such as attribute selection and boosting, we can construct a powerful and reliable predictive model. This comprehensive approach harnesses the strength of various techniques to create a highly effective and accurate solution.

BIBLIOGRAPHY

WEBSITE

1.) https://www.w3schools.com/

2.) https://www.wikipedia.org/

3.) https://www.google.co.in/

4.) https://www.ChatGPT.com/

5.)https://www.geeksforgeeks.org/

6.)https://www.javatpoint.com/

7.)https://colab.research.google.com/

REFERENCE Article / Pages

1) Swapna, B., Manivannan, S., & Kamalahasan, M. (2022). Prognostic of soil nutrients and soil fertility index using machine learning classifier techniques. International Journal of e-Collaboration (IJeC), 18(2), 1-14.

2) Janvier, N. I. Y. I. T. E. G. E. K. A., Arcade, N., Eric, N. G. A. B. O. Y. E. R. A., & Jean, N. (2021). Machine learning based soil fertility prediction. International Journal of Innovative Science, Engineering & Technology, 8(7), 141-146.

3) Koley, S. (2014). Machine learning for soil fertility and plant nutrient management using back propagation neural networks. Shivnath Ghosh, Santanu Koley (2014)"Machine Learning for Soil Fertility and Plant Nutrient Management using Back Propagation Neural Networks" International Journal on Recent and Innovation Trends in Computing and Communication, 2(2), 292-297.

4) Keerthan Kumar, T. G., Shubha, C. A., & Sushma, S. A. (2019). Random forest algorithm for soil fertility prediction and grading using machine learning. Int J Innov Technol Explor Eng, 9(1), 1301-1304.

5) Janmejay Pant, Pushpa Pant, R. P. Pant, Ashutosh Bhatt, Durgesh Pant, Amit Juyal, Soil Quality Prediction for Determining Soil Fertility in Bhimtal Block of Uttarakhand (India) Using Machine Learning, Int. J. Anal. Appl., 19 (1) (2021), 91-109.

6) "KAGGLE,"https://www.kaggle.com/datasets/rahuljaiswalonkaggle/soil-fertility-dataset .

7) H. Zheng, J. Wu, and S. Zhang, “Study on the spatial variability of farmland soil nutrient based on the kriging interpolation,” 2009 Int. Conf. Artif. Intell. Comput. Intell. AICI 2009, vol. 4, pp. 550–555, 2009, doi: 10.1109/AICI.2009.137.

8) Prince Patel, “Why Python is the most popular language used for Machine Learning,” 2018. [Online]. Available: https://medium.com/@UdacityINDIA/why-use-python-for-machine learning-e4b0b4457a77. [Accessed: 20-Mar-2020].

9) N. Gupta, “Why is Python Used for Machine Learning?,” 2019. [Online].Available:https://hackernoon.com/why-python-used-for-machine learning-u13f922ug. [Accessed: 20-Mar-2020].

10) “Soil test”,Wikipedia,June 2012.

11) Vamanan R. & Ramar K. (2011) “CLASSIFICATION OF GRICULTURAL LAND SOILS A DATA MINING APPROACH”; International Journal on Computer Science and Engineering (IJCSE); ISSN: 0975-3397 Vol. 3

12) Witten I. and Eibe F. (2005), “Data Mining: Practical Machine Learning Tools and Techniques” 2nd Edition, San Francisco: Morgan Kaufmann