Project Report

Practice School Internship

Hands-on "AI Based Server and it's Language Independent communication mechanism with External Systems"

Submitted By

BODASINGI SANDEEP

2000080015

Bachelor of Technology

In

Artificial Intelligence & Data Science Department

KL Deemed to be University

Guntur (Andhra Pradesh) - 522502

Bonafide Certificate

The Project submitted by **BODASINGI SANDEEP (College ID No: 2000080015)** on "**AI Based Server and its Language Independent Communication Mechanism with External Systems**" in fulfilment of the requirement of **Practice School Internship for the award of the degree of B. Tech in Artificial Intelligence and Data Science at K L University, Vaddeswaram, Guntur District, Andhra Pradesh** is a bonafide record of work carried out **at Electronics and Radar Development Establishment (LRDE) DRDO, Bangalore from August-2023 to October-2023** under our supervision and guidance.

[illegible]

(Rohit Ganeshan)
Scientist 'C'
D-SR, LRDE
Bangalore – 560093

K L Deemed to be UNIVERSITY

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

Certificate

This is Certified that the project entitled **“AI Based Server and it’s Language Independent communication mechanism with External Systems”** which is a experimental work carried out by Sandeep Bodasingi (2000080015), in partial fulfillment for the award of the degree of **Bachelor of Technology** in **Department of Artificial Intelligence And Data Science** , during the year **2023-24**. The project has been approved as it satisfies the academic requirements.

Department Coordinator **Head of the Department**

University Guide **External**

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to the Board of Management of KL University for their kind encouragement in doing this practice school and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. Badugu Suresh** incharge of the practice school, and **Dr. Gandharba Swain** Head of Computer Science and Engineering for providing me with necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Mr. Rohit Ganeshan Scientist 'C' at LRDE(DRDO)** for his valuable guidance, suggestions, and constant encouragement that paved the way for the successful completion of my project work.

I wish to express my thanks to all the teaching and non-teaching staff members of the Department of Artificial Intelligence And Data Science were helpful in many ways for the completion of the project.

About the Organisation

The Electronics and Radar Development Establishment (LRDE) is one of the premier R&D Establishments set up under the Defence R&D Organisation. Established in the year 1958,LRDE is committed to the indigenous Development of Modern Radar Systems through an effective Quality Management System.

Vision of LRDE:

To create a center of excellence in Radar and Electronics Technologies by developing and delivering world class systems to meet the customer requirements.

Mission of LRDE:

- To design and develop Radar systems meeting the current and futuristic requirements of Services, keeping in view the emerging threat and EW scenario.
- To contribute towards the establishment of indigenous production of our designs through public and private sector, work committedly towards introduction of the systems into Services.
- To promote research and competence building activities in the field of Radar within the laboratory and in academic institutions and continuously evolve as the center of excellence in Radar Technology.

Charter of Duties:

- Develop Radar Systems for Indian Army, Navy, Air Force and other paramilitary organisation.
- Development of Radar Technologies to develop state-of-the-art radars.
- Develop Core Competence in contemporary and futuristic radar technologies.
- Expert advice and consultancy to Services and Allied Forces towards collaborative development of radar systems.
- Support programs for Electronic components production, quality assurance and certification.

About the Internship

As an intern at LRDE, I worked on Machine Learning Algorithms. I learnt the brief process of how a RADAR works and its uses.

During my internship with the organisation, I learned about RADARs, Communication of RADARs in brief, Artificial Intelligence and Machine learning Algorithms used in RADARs and in its communication.

I was able to put these skills in practise and the experience I got from working here is far more learning than the theory knowledge I had. I was given the opportunity to put my skills at work and I am thankful for giving me the chance.

I felt that their mission was compatible with my interests and my future career goals, which is the reason I applied for an internship position in this organisation.

Table of Contents

Chapter 1:

Introduction

This project report contains the details of what I have learnt throughout the three months' time period. The main aim of the project was to facilitate communication with RADARs by utilizing a Support Vector Machine (SVM) for data analysis, AI Model training for classification and UDP (User Datagram Protocol) programming for real-time data exchange.

In this project, we have implemented a Support Vector Machine (SVM) for the Iris dataset, split the data into training and testing sets, hosted the trained SVM model on a UDP server, and developed an AI server for accepting real time data from external systems as input. Additionally, we have created a client program to send data to the AI server and receive responses. This client system program demonstrates the way in which external systems can connect to AI Server.

Radar systems are indispensable in various industries, providing crucial data for decision-making, monitoring, and safety. To further improve communication with RADARs, our project aims to leverage machine learning techniques for data analysis and UDP programming for real-time data exchange.

Chapter 2:

RADAR

Introduction to RADARS:

- RADAR is an electromagnetic sensor which used Electromagnetic Waves (EM Waves) to track, locate, and measure various aspects of the object which are at a distance.
- RADAR sends out EM Waves through antenna and these waves strike the objects present in air and reflects back to the antenna.
- RADAR signals can target several objects at the same time.

Components of RADAR:

Antenna and Duplexer:

The antenna can be either a mechanically rotating reflector or a phased array electronically steered in azimuth and elevation.

Duplexer helps the signal antenna for transmission and reception. It also helps in protecting the sensitive receiver from the high power transmitter.

Waveform Generator:

It tailors the waveform to the environment and to the particular operating mode actually used.

Signal Processor:

It determines the presence or absence of targets while rejecting unwanted signals due to ground clutter, sea clutter, weather, radio-frequency, noise sources, and intentional jammers.

Data Extractor:

This provides the target measurements in range, angles(azimuth and elevation), radial speed, etc.

In general, targets may cause several detections in adjacent cells in range.

Data Processor:

It is essentially where tracking filtering is implemented.

User:

The output is generally displayed to visualize the information contained in the radar echo signal in a form suitable for operator interpretation and action.

The plan position indicator (PPI) is deployed in radar which indicates the range and azimuth of a detected target.

The visualized information on the display is called the synthetic video.

Controller:

This decodes commands from the operator and sets up the operation modes, the appropriate system timing and the signal generator together with the processing functions on the received signals according to range, azimuth, and elevation sectors.

The controller also analyses signals for fault detection. It normally comprises a set of software programs implemented on a digital computer.

Communication between RADAR Components:

Computer Networks:

A computer Network is a set of devices connected through links. A node can be a computer, printer, or any other device capable of sending or receiving the data.
The links connecting the nodes are called the communication channels. Computer Network uses Distributed Processing in which the tasks are divided among several computers.

Protocol:

It is a set of rules for formatting and processing the data.
The computers within a network use vastly different software and hardware but the use of protocols enables them to communicate with each other regardless.

Datagram:

It is an independent, self-contained message sent over the network whose arrival. Arrival time and content are not guaranteed. It refers to the smallest unit via the data is transmitted. Datagrams are data packets which contain adequate header information so that they can be individually routed by all intermediate network switching devices to the destination.

UDP:

- The UDP Protocol allows the computer applications to send the messages in the form of datagrams from one machine to another over Internet Protocol (IP).

- UDP works by encapsulating the data into the packet and providing its own header information to the packet. Then this UDP packet is encapsulated to the IP packet and sent off to its destination.

- Since UDP sends the messages in the form of datagrams, it is considered as the best-effort mode of communication.

- UDP also provides a different port number to distinguish different user requests and also provide the checksum capability to verify whether the complete data has arrived or not.

TCP:

- It is designed to send packets across the internet and ensure the successful delivery of data and messages over network.

- TCP organises data so that it can be transmitted between a server and client. It guarantees the integrity of the data being communicated over a network.

- TCP is comparatively slower than UDP. Retransmission of lost packets is possible in TCP

RDP:

- ➢ RDP is short form for Radar Data Processor.The primary function of RDP is to convert Signal Processor Plot Data related to target detections into target tracks.
- ➢ In particular, the image must be constructed to ensure that a trained user can make maximum use of the information that is available within the returned radar signals.

Problem Statement for the project:

Problem Statement:

AI Module Implementation in SVM and its Interconnection with External systems in a language Independent manner.

Procedure:

Step 1: Data Preparation

- Import the Iris dataset.
- Split the dataset into two parts: 80% for training and 20% for testing.
- Save both the training and testing datasets in separate files.
- Once the module was tested with IRIS data, the same process was used for Radar Track data also.

Step 2: SVM Model Training

- Load the 80% training data from the saved file.
- Implement the SVM algorithm using Python libraries (e.g., Scikit-Learn).
- Train the SVM model using the training data.
- Save the trained SVM model to a file for future use.

Step 3: UDP Server Setup

- Create a Python UDP server that will host the SVM model.
- Load the SVM model from the saved file.
- Configure the server to listen on a specific UDP port for incoming data.

Step 4: Testing with Test Data

- Read the 20% test data from the saved file.
- Establish a connection to the UDP server.
- Send each data point from the test data to the UDP server line by line for classification.
- Receive and collect the classification results from the server.

Step 5: AI Server Development

- Create an AI server using Python.
- Configure the AI server to accept live data from external systems over UDP Port.
- Implement logic to process incoming data and use the loaded SVM model for classification.

Step 6: Client Program

- Develop a client program in c language for interaction with the AI server.
- The client program should accept user input and send it to the AI server.
- Receive and display the responses from the AI server.
- Include logic to wait for the next input and handle user interactions.

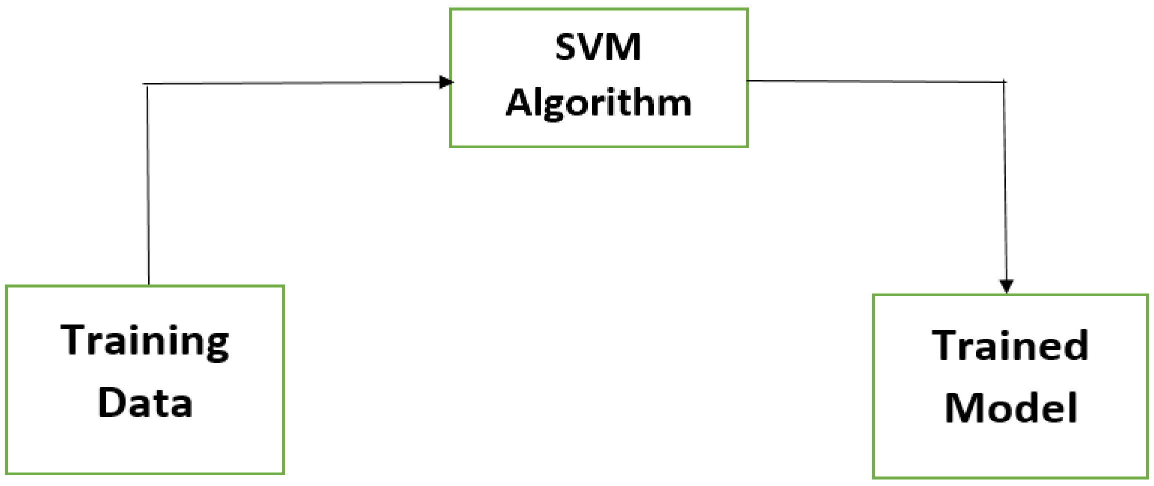
Step 7: Logging and Monitoring

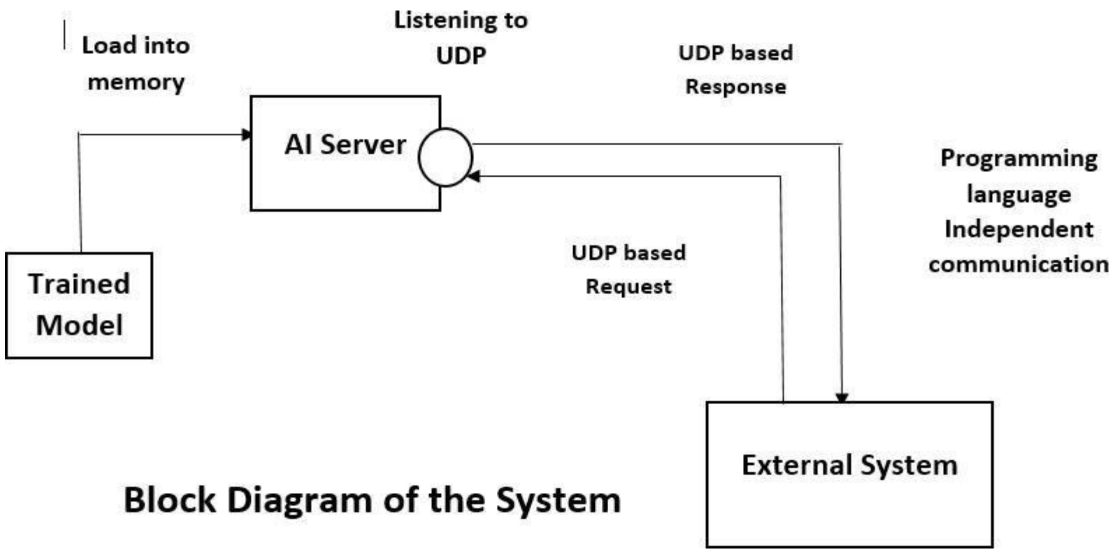
- Implement a logging system in both the UDP server and the AI server. Log.
- Important information such as port numbers, client addresses, and requests.
- This information helps with debugging and monitoring the communication with RADARs .

Step 8: Testing and Validation

- Thoroughly test the entire system to ensure it functions as expected. Validate the SVM model's accuracy using the 20% test data.
- Test the client program's usability and responsiveness.

Block Diagram :





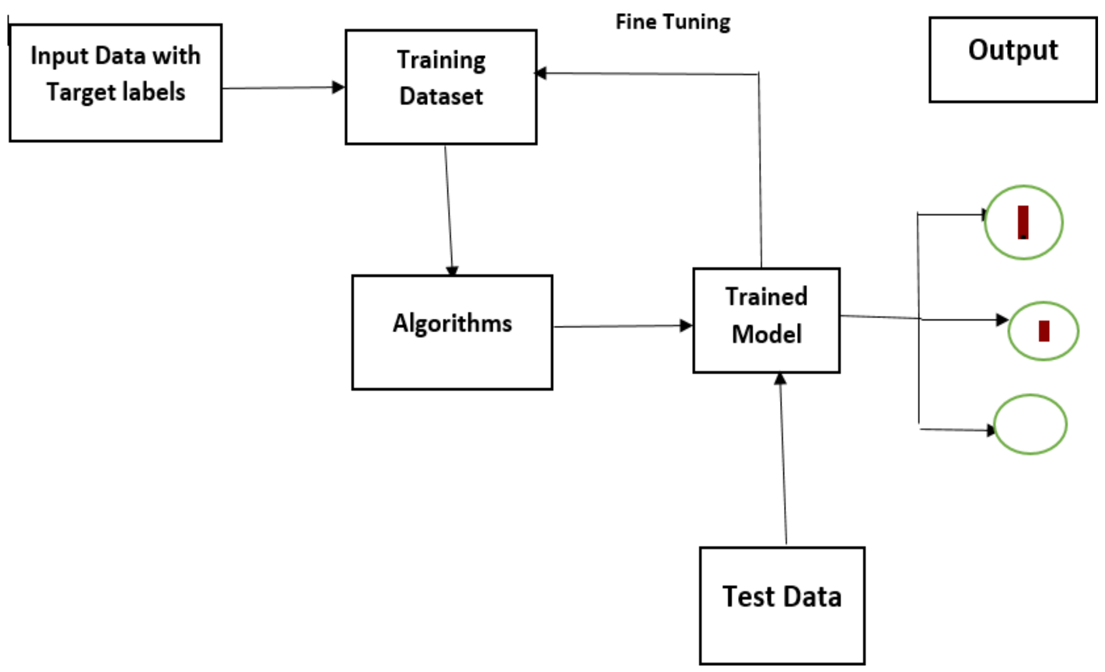
Block Diagram of the System

Chapter 3:

Artificial Intelligence and Machine Learning

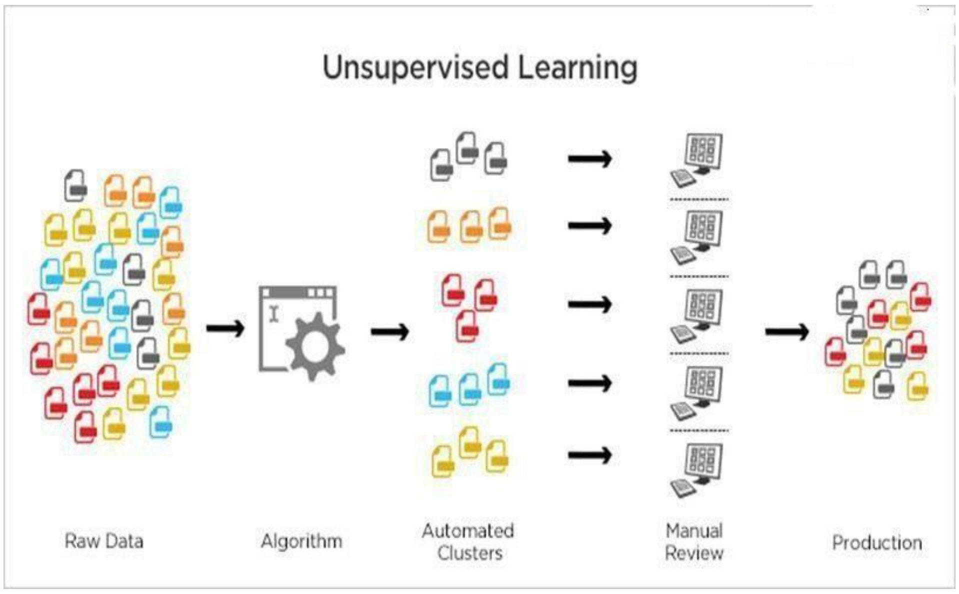
Supervised learning:

- Supervised learning involves using a labeled dataset where each data point has both input features and corresponding output labels.
- The primary objective is to train a model to make accurate predictions or classifications based on the input features.
- Supervised learning can be divided into two main types: classification (for categorical outputs) and regression (for continuous outputs).
- During training, the algorithm learns from the labeled data by adjusting its internal parameters to minimize prediction errors, typically using optimization techniques and loss functions.



Unsupervised learning:

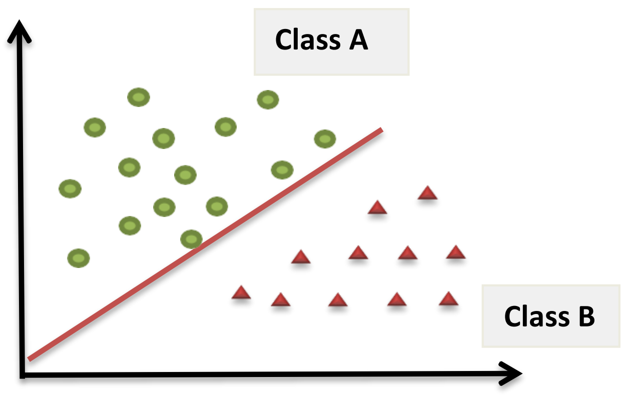
- Unsupervised learning deals with datasets that lack explicit output labels or target variables.
- The primary objectives are to discover patterns and structure within the data, such as grouping similar data points (clustering) or reducing the complexity of data (dimensionality reduction).
- Unsupervised learning includes techniques like clustering algorithms (e.g., K-Means) and dimensionality reduction methods (e.g., Principal Component Analysis - PCA).
- Another application of unsupervised learning is identifying anomalies or outliers in the data, which are data points that deviate significantly from the norm.



Classification:

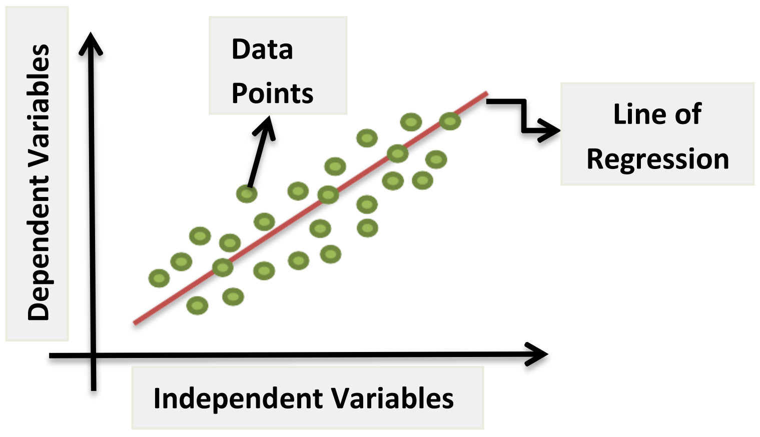
- It is a supervised learning algorithm for finding categories of observations based on training data.
- Here, the algorithm learns from the training dataset and categorizes the observations to classes or groups.
- In Classification, the output variable is a category and not a value.
- An algorithm which implements classification on dataset is “classifier”. Types of classifiers:
  1) BinaryClassifier – Yes/No, Male/Female, True/False

2) Multi Class Classifier – Colour of the ball out of Yellow, Blue , Green and Red.



Regression:

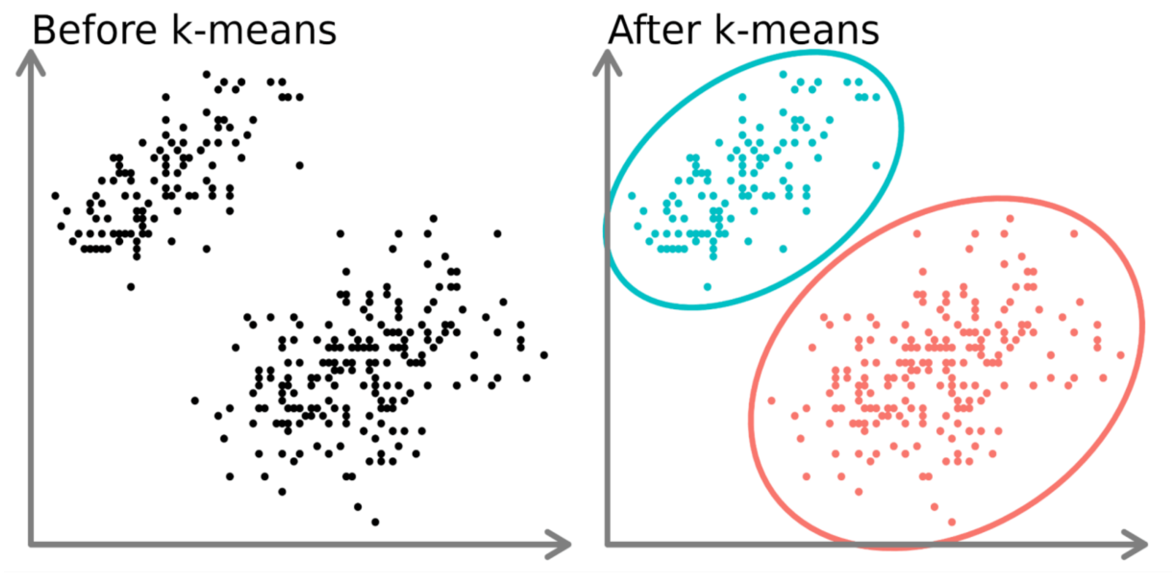
- It is a process to find the correlation between dependent and independent variable.
- It predicts continuous variables like Stock prices, etc.
- Regression algorithm aims to find a mapping function to input variable (X) with continuous output variable (Y).



K-Means Clustering:

- K-Means is an unsupervised machine learning algorithm used for clustering or grouping data points based on their similarity.
- K-Means uses a centroid-based approach, where it divides the data into 'k' clusters, with each cluster represented by a centroid (center point).
- The algorithm aims to minimize the sum of squared distances between data points and their assigned cluster centroids.
- It iteratively updates the centroids and assigns data points to the nearest centroid until convergence.

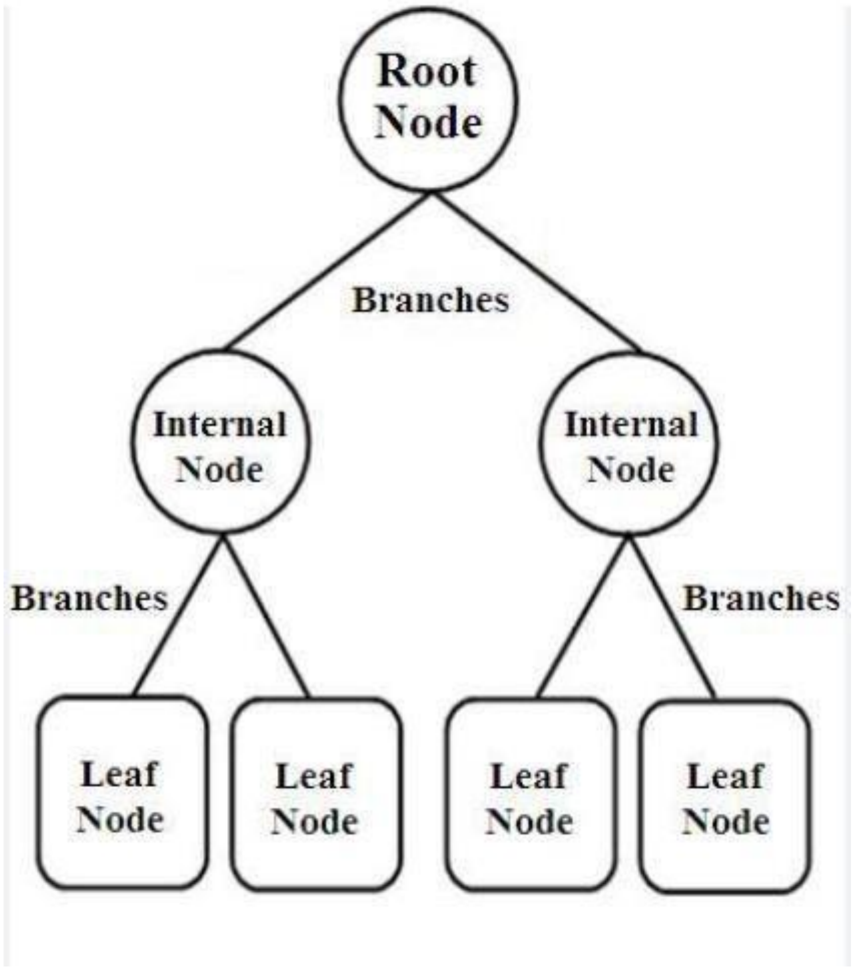
- The number of clusters 'k' is a critical parameter that must be specified before applying K-Means.
- The choice of 'k' can impact the quality of clustering results, and various methods can be used to determine the optimal 'k' value.



Decision Tree:

- A decision tree is a supervised machine learning algorithm used for both classification and regression tasks.
- It represents decisions and their possible consequences in a tree-like structure, where each node represents a decision or a test on a feature, and each branch represents an outcome or decision.
- The tree starts at the root node, and at each node, a decision or test is made based on a feature.
- Depending on the outcome of the test, the tree traverses down to child nodes, continuing the decision-making process until a leaf node is reached, which provides the final prediction or outcome.
- Decision trees use various criteria (e.g., Gini impurity, information gain, or mean squared error) to determine the best feature and value to split the data at each node.
- The goal is to minimize impurity (for classification) or variance (for regression) and maximize the homogeneity of data in child nodes.

- However, they are prone to overfitting, where the model captures noise or details specific to the training data. Pruning or limiting tree depth can help mitigate overfitting.



Support Vector Machine (SVM):

- Support Vector Machine (SVM) is a supervised machine learning algorithm used for both classification and regression.
- Though we say regression problems as well it's best suited for classification.
- The main objective of the SVM algorithm is to find the optimal hyperplane in an N-dimensional space that can separate the data points in different classes in the feature space.
- The hyperplane tries that the margin between the closest points of different classes should be as maximum as possible.
- The dimension of the hyperplane depends upon the number of features. If the number of input features is two, then the hyperplane is just a line.

➢ If the number of input features is three, then the hyperplane becomes a 2-D plane. It becomes difficult to imagine when the number of features exceeds three.

➢ **Types of Support Vector Machine :**

Linear SVM: Linear SVMs use a linear decision boundary to separate the data points of different classes. When the data can be precisely linearly separated, linear SVMs are very suitable.

Non-Linear SVM: Non-Linear SVM can be used to classify data when it cannot be separated into two classes by a straight line (in the case of 2D). By using kernel functions, nonlinear SVMs can handle nonlinearly separable data.

➢ **Advantages of SVM:**

- Effective in high-dimensional cases.
- Its memory is efficient as it uses a subset of training points in the decision function called support vectors.
- Different kernel functions can be specified for the decision functions and its possible to specify custom kernels.

Implementation of SVM Algorithm:

a) Data Preparation:

- Begin by preparing your dataset, ensuring it is labeled and appropriately formatted for classification.
- Split the data into training and testing sets to assess model performance.

b) Choose Kernel Function:

- Select a suitable kernel function based on the nature of your data (e.g., linear, polynomial, RBF).
- The kernel function determines how data is mapped to a higher-dimensional space.

c) Model Initialization:

- Initialize an SVM model with the chosen kernel function and optional hyperparameters (e.g., regularization parameter C).

d) Model Training:

- Train the SVM model using the training data.

- The model aims to find the hyperplane that maximizes the margin between classes while minimizing misclassifications.

e) <u>Hyperparameter Tuning:</u>

- Experiment with different hyperparameter values to find the optimal configuration that balances margin size and classification accuracy.
- Techniques like cross-validation can help in hyper parameter tuning.

f) <u>Support Vector Identification:</u>

- Identify the support vectors, which are the data points closest to the decision boundary.
- These support vectors are essential for defining the margin and decision boundary.

g) <u>Model Evaluation:</u>

- Evaluate the SVM model's performance using the testing data.
- Common metrics for classification tasks include accuracy, precision, recall, F1-score, and ROC curves.

h) <u>Visualization (Optional):</u>

- Visualize the decision boundary and support vectors if your data is in a two-dimensional space.
- Visualization can provide insights into how the model classifies data.

i) <u>Deployment:</u>

- Once satisfied with the model's performance, deploy it to make predictions on new, unseen data.

j) <u>Interpretation:</u>

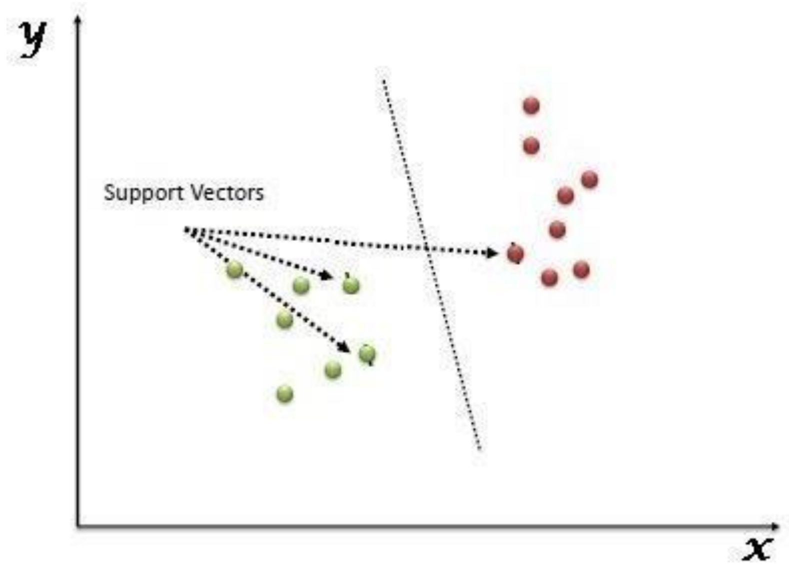
- Interpret the model's decision boundary and support vectors to gain insights into how it makes predictions.
- SVM models are known for their transparency and interpretability.

k) <u>Regularization (Optional):</u>

- Consider using regularization techniques like soft-margin SVM to handle noisy or overlapping data.
- The regularization parameter C controls the trade-off between margin size and misclassification.

l) <u>Scaling (Optional):</u>

- Depending on the kernel function and data, it may be necessary to scale or normalize the input features for better model performance.



➢ These steps provide a high-level overview of the implementation of the SVM algorithm. The specific implementation details and choice of libraries (e.g., Scikit-Learn in Python) will depend on the programming language and tools you're using for your project.

Chapter 4:

Dataset used in this project

Fisher's Iris Data Set:

The Fisher's Iris dataset is a famous dataset in machine learning and statistics. Here are some key points about it:

- The dataset was introduced by British biologist and statistician Ronald A. Fisher in 1936 as an example of discriminant analysis.
- **Data:** It contains measurements of 150 iris flowers from three different species: Setosa, Versicolor, and Virginica. There are four features measured for each flower:
  - Sepal length (in centimeters)
  - Sepal width (in centimeters)
  - Petal length (in centimeters)
  - Petal width (in centimeters)

Objective: Fisher used this dataset to illustrate how different species of iris flowers could be distinguished based on their measurements.

- **Use in Machine Learning:** The Iris dataset is often used as a beginner's dataset for classification tasks, particularly in machine learning and data science courses.
- **Distribution:** It is readily available and can be accessed from various sources and libraries in different programming languages like Python.
- **Sample Size:** The dataset consists of 150 samples, with 50 samples from each of the three species.
- **Species Classes:**
  - Iris Setosa
  - Iris Versicolor
  - Iris Virginica

Analysis: The dataset can be analyzed to understand the differences in the measurements between the three species, and various classification algorithms can be applied to predict the species based on themeasurements.

- **Visualization:**Scatterplots, histograms, and other visualization techniques are often used to explore and visualize the data.

- **Machine Learning Models:** Common machine learning algorithms like k-nearest neighbors, decision trees, and support vector machines are often used to classify iris flowers based on their features.
- **Dataset Availability:** The dataset is readily available in many machine learning libraries, including scikit-learn in Python.
- **Accuracy Benchmark:** Due to its simplicity and well-separated classes, it serves as a benchmark for assessing the performance of classification models.
- **Data Integrity:** It is considered a clean dataset with no missing values or outliers, making it suitable for educational purposes and model testing.
- **Named After:** The dataset is named after the iris flowers, and it's sometimes referred to as Fisher's Iris data or Fisher's Iris dataset in honor of its creator.
- **Widely Used:** Despite its age, the Iris dataset remains a popular choice for teaching, testing, and benchmarking machine learning algorithms.

After the system was successfully tested with Fisher’s Iris Dataset, The same was tested with Radar Track Data Set as well.

Chapter 5:

Python and Socket Programming Concepts used in this project

Python packages and modules:

- **Scikit-learn :** scikit-learn is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support-vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.
- **Pandas :** Pandas is an open-source library in Python that is made mainly for working with relational or labeled data both easily and intuitively. It provides various data structures and operations for manipulating numerical data and time series. This library is built on top of the NumPy library of Python. Pandas is fast and it has high performance & productivity for users.
- **Joblib :** Joblib is a Python library used for lightweight pipelining in Python. It provides tools for saving and loading Python objects, particularly for objects that are computationally expensive to create. joblib is often used for caching and efficient handling of objects like machine learning models, NumPy arrays, and other large data structures.
- **Numpy :** NumPy is 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. The predecessor of NumPy, Numeric, was originally created by Jim Hugunin with contributions from several other developers. NumPy is open-source software and has many contributors. NumPy is a NumFOCUS fiscally sponsored project.
- **Socket :** The socket module is used for socket-based network communication. It allows the creation of network sockets and provides

functions to send and receive data over the network using various protocols. In this code, it is used to create a UDP server, receive data from clients, and forward data to another system.

Client-Server Application:

- A client-server application is a program that runs on the client-side while accessing the information over a remote server. The client-server always makes requests to the remote server by calling functions of the server to retrieve information. The client program and the server program may run on different systems and on different platforms where they require a communications layer known as middleware.
- The client-server application might run on a network client or a network server. The applications are solely described on their architecture except for the fact that how it is deployed on the network. It uses a two-tier architecture that has two users; the client and the server.
- The server machine acts as a host that can run single or multiple server programs that share their resources with the clients. Sometimes the server gets overloaded when simultaneous requests are received from the client.
- The communication between the client and server is a two-way street. Servers can reach the client to make sure that the client has appropriate updates, patches, or if there are any other requirements. Once the inquiry is done, the server closes the connection to the client so that the bandwidth space and the network are conserved.
- Some of the typical features of client-server applications are as follows;
    - Multiple client programs have the ability to request services from a single server.
    - A single client program can request services from multiple server programs.
    - A single server program has the ability to provide multiple services.
    - The client program doesn’t have to be aware of the number of subprograms that provide a service.
    - Multiple subprograms have the ability to work together to provide a service.

- The server programs run on a machine that is remote from the machine that runs the client program.

Fundamental Concepts in C Programming :

- **Header Files Inclusion:**
  - ‘#include <stdio.h>’ : Includes the standard input/output library for functions like printf and scanf.
  - ‘#include <stdlib.h>’ : Includes the standard library, providing functions like malloc and free.
  - ‘#include <string.h>’ : Includes the string manipulation library, used for functions like strcmp and strlen.
  - ‘#include <winsock2.h> ’: On Windows, this includes the Winsock 2 library, which is essential for socket programming.
- **Function Declarations:**
  - ‘int main()’ : Defines the main function, which serves as the entry point for the program.
- **Variable Declarations and Initialization:**
  - ‘WSADATA wsaData;’ : Declares a structure (‘wsaData’) used to store information about Winsock initialization.
  - ‘SOCKET clientSocket;’ : Declares a socket (‘clientSocket’) for communication.
  - ‘struct sockaddr_in serverAddr;’ : Declares a structure (‘serverAddr’) to represent the server's address information.
  - ‘char serverIP[] = "127.0.0.1";’ : Initializes a character array (‘serverIP’) with the server's IP address.
  - ‘int serverPort = 12345;’ : Initializes an integer (‘serverPort’) with the server's port number.
  - ‘char testingData[1024];’ : Declares a character array (‘testingData’) to store user input.
- **Winsock Initialization:**
  - ‘WSAStartup(MAKEWORD(2, 2), &wsaData)’ : Initializes the Winsock library, specifying version 2.2. This is required for socket operations on Windows.

- **<u>Socket Creation:</u>**
  - ‘socket(AF_INET, SOCK_DGRAM, 0)’ : Creates a UDP socket (‘clientSocket’) for communication.
- **<u>Socket Address Configuration:</u>**
  - ‘serverAddr.sin_family’ : Specifies the address family (IPv4).
  - ‘serverAddr.sin_port’ : Sets the port number to connect to.
  - ‘serverAddr.sin_addr.s_addr’ : Sets the IP address to connect to (‘serverIP’).
- **<u>User Interaction:</u>**
  - ‘while (1)’ : Initiates an infinite loop to continuously read user input.
  - ‘fgets(testingData, sizeof(testingData), stdin)’ : Reads a line of input from the user and stores it in testingData.
  - ‘strcmp(testingData, "exit\n") == 0’ : Compares the input with "exit" to exit the loop if the user enters "exit."
- **<u>Data Sending:</u>**
  - ‘sendto(clientSocket, testingData, strlen(testingData), 0, (struct sockaddr*)&serverAddr, sizeof(serverAddr))’ : Sends data to the server using the ‘sendto’ function.
- **<u>Socket Cleanup:</u>**
  - ‘closesocket(clientSocket)’ : Closes the socket when communication is complete.
  - ‘WSACleanup()**’ :** Cleans up Winsock library resources when the program exits.

Chapter 6:

Implementation

Question : Write a program to split the Iris dataset into 2 parts as 80% for training the SVM and 20% for testing. Both 80% and 20% should be in different files.

Solution :

```
pip install scikit-learn pandas
 import pandas as pd

from sklearn import datasets

from sklearn.model_selection import train_test_split

from sklearn.svm import SVC

from sklearn.metrics import accuracy_score

import joblib


iris = datasets.load_iris()

X =  iris.data

y = iris.target


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)


svm_classifier = SVC(kernel='linear', C=1)


svm_classifier.fit(X_train, y_train)
```

```
y_pred = svm_classifier.predict(X_test)


accuracy = accuracy_score(y_test, y_pred)

print(f"Accuracy: {accuracy:.2f}")


joblib.dump(svm_classifier, 'svm_model.pkl')


train_data    =    pd.DataFrame(data=X_train,    columns=iris.feature_names)

train_data['target'] = y_train

test_data    =    pd.DataFrame(data=X_test,    columns=iris.feature_names)

test_data['target'] = y_test

train_data.to_csv('train_data.csv', index=False)

test_data.to_csv('test_data.csv', index=False)
```

Output :

```
SVC(C=1, kernel='linear')
```

Question : Write a program for implementation of SVM algorithm using the tra ining data file(80% data) and save the trained model in a file.

Solution:

```
import pandas as pd
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
import joblib

train_data = pd.read_csv('train_data.csv')

X_train  =  train_data.drop('target',  axis=1)
y_train = train_data['target']
```

```
svm_classifier = SVC(kernel='linear', C=1)

svm_classifier.fit(X_train, y_train)

test_data = pd.read_csv('test_data.csv')

X_test = test_data.drop('target', axis=1)
y_test = test_data['target']

y_pred = svm_classifier.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)

print(f"Accuracy: {accuracy:.2f}")

joblib.dump(svm_classifier, 'svm_model_trained_on_80_percent.pkl')
```

Output:

```
['svm_model_trained_on_80_percent.pkl']
```

Question : Write a program to read the 20% (Test Data) and pass it to SVM model for testing.

Solution:

```
import pandas as pd
from sklearn.metrics import accuracy_score
import joblib

svm_model = joblib.load('svm_model_trained_on_80_percent.pkl')

test_data = pd.read_csv('test_data.csv')

X_test = test_data.drop('target', axis=1)
y_test = test_data['target']

y_pred = svm_model.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy on testing data: {accuracy:.2f}")
```

Output:
Accuracy on testing data: 98%

Question: Write a Client program so that SVM model (from 80% data) is loaded and hosted on a UDP port. Send Data line by line from 20%(Test file) to UDP Port using UDP Programming in C language.

Solution:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <winsock2.h>

int main() {
    WSADATA wsa;
    SOCKET client_socket;
    struct sockaddr_in server_addr;
    char server_ip[] = "127.0.0.1";
    int server_port = 12345;

    if (WSAStartup(MAKEWORD(2, 2), &wsa) != 0) {
        perror("Failed to initialize Winsock");
        return 1;
    }

    if ((client_socket = socket(AF_INET, SOCK_DGRAM, 0)) == INVALID_SOCKET)
{
        perror("Failed to create socket");
        return 1;
    }

     server_addr.sin_family = AF_INET;
     server_addr.sin_port = htons(server_port);
     server_addr.sin_addr.s_addr = inet_addr(server_ip);

     while (1) {
            char testing_data[1024];
            printf("Enter comma-separated numeric features : ");
            fgets(testing_data, sizeof(testing_data), stdin);
            testing_data[strcspn(testing_data, "\n")] = '\0';

            if (sendto(client_socket, testing_data, strlen(testing_data), 0, (struct
sockaddr*)&server_addr, sizeof(server_addr)) == SOCKET_ERROR) {
                perror("Failed to send data");
                return 1;
           }
          char server_response[1024];
          int server_response_len;
          if ((server_response_len = recvfrom(client_socket, server_response, size
of(server_response), 0, NULL, NULL)) == SOCKET_ERROR) {
              perror("Failed to receive response");
              return 1;
          }
          server_response[server_response_len] = '\0';
          printf("Server response: %s\n", server_response);
    }
    closesocket(client_socket);
    WSACleanup();

    return 0;
}
```

Output :

```
Enter comma-separated numeric features : 1.2, 6.3, 2.3, 6.3
Predicted Class: 2, Client Port: 62852
Enter comma-separated numeric features :
```

Question: Write a server program so that the AI server accepts comma separated Values and prints the response, port number and waits for the next input in python .

Solution :

```
import  joblib
import socket
import numpy as np

svm_model = joblib.load('svm_model_trained_on_80_percent.pkl')

server_ip = '127.0.0.1'
```

```
server_port = 12345

server_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
server_socket.bind((server_ip, server_port))

print(f"UDP server is listening on {server_ip}:{server_port}")

while True:
    data, addr = server_socket.recvfrom(1024)
    data = data.decode('utf-8')

    client_ip, client_port = addr

    try:
        data = [float(x) for x in data.split(',')]

        result = svm_model.predict([data])
        response = f"Client Port: {client_port}, Request: {data}, Predicted Cla
ss: {result[0]}"

        print(response)
    except (ValueError, IndexError):
        response = "Invalid data format. Please provide comma-separated nu
meric features."

    server_socket.sendto(response.encode('utf-8'), addr)
```

Output :

```
UDP server is listening on 127.0.0.1:12345
Client Port: 62852, Request: [1.2, 6.3, 2.3, 6.3], Predicted Class: 2
```

Chapter 7:

Conclusion

This project represents a sophisticated AI-based classification system that seamlessly integrates machine learning, network programming, and real-time data exchange for improved communication with RADARs. The process begins with the implementation of a Support Vector Machine (SVM) algorithm for classification using the Iris dataset as well as Radar Dataset, thoughtfully partitioned into distinct training and testing datasets. The trained SVM model is then persistently saved for subsequent usage. Building on this, the project demonstrates a robust real-time communication infrastructure by hosting the SVM model on a UDP server. This server efficiently receives data line from the 20% testing dataset, illustrating a powerful mechanism for RADAR communication. Furthermore, a client program simplifies user interactions with the AI server, providing an intuitive interface for sending and receiving requests while offering clarity on port numbers, client addresses, and requests in the server's output. By seamlessly uniting these components, this project representsa significant leap toward streamlining communication with RADARs, offering not only an adaptable framework for RADAR-related tasks but also showcasing the potential of AI-driven communication solutions in broader applications. Additionally, the use of AI server programmed in python and external system (client program) developed in C language demonstrated a language independent communication mechanism.