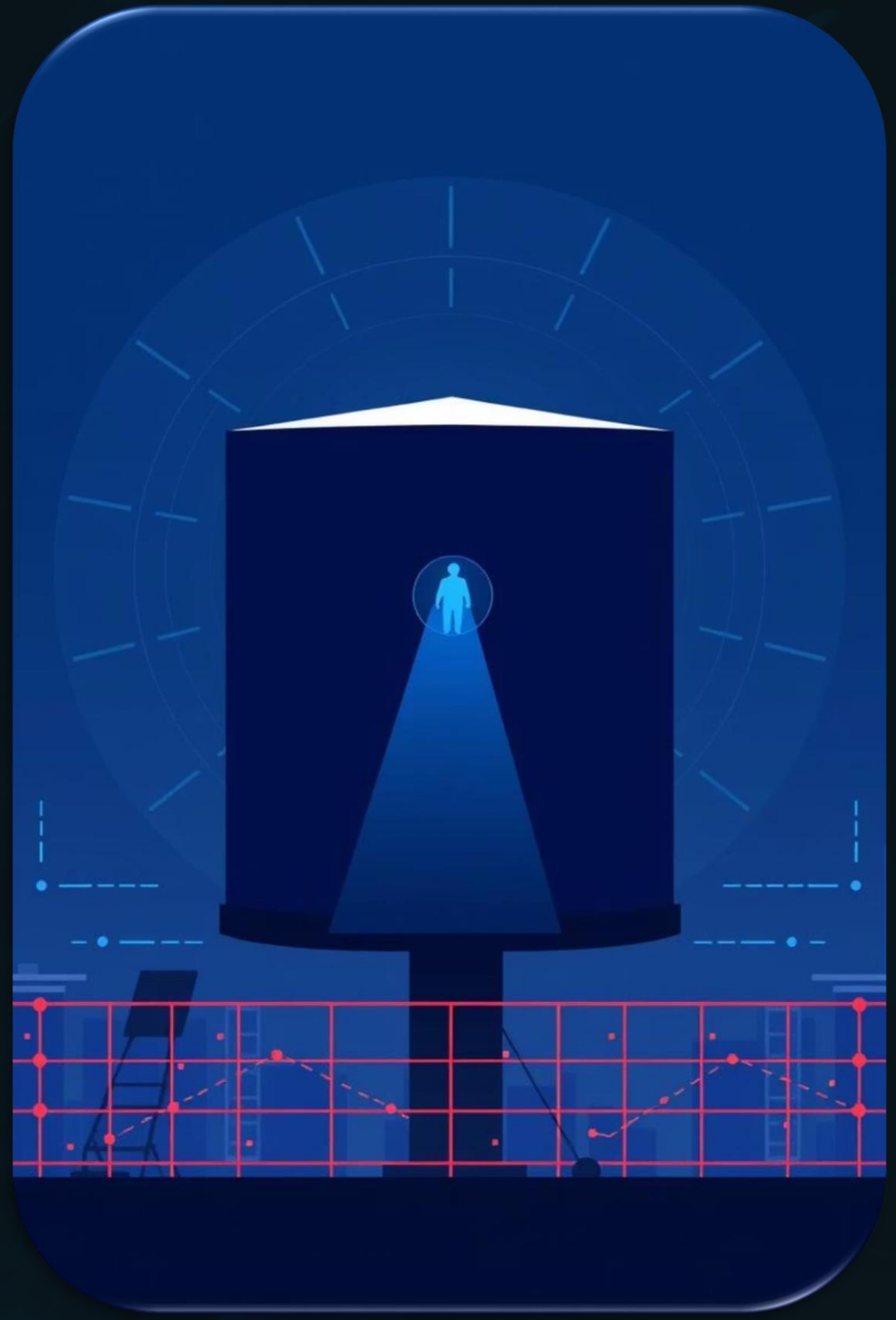
Radar-Based Intrusion Detection with Data Analytics

Building Smart Perimeter Monitoring Systems Using mmWave Technology

Key Focus Areas:

- *Advanced radar signal processing*
- *Machine learning classification*
- *Real-time intrusion detection*
- *Data-driven security solutions*

Naman Malhotra
Vrishank Sharma
Ritika Gupta
Github Link: [Intrusion Detection Project Github Link]
Institute: Maharaja Agrasen Institute of Technology



Why Radar-Based Intrusion Detection?

The increasing demand for robust and reliable perimeter security systems highlights a critical need for advanced detection technologies. Traditional surveillance methods, while useful, often fall short in challenging conditions.

- Traditional security systems face significant limitations in various environmental conditions. Cameras struggle with **poor lighting and weather**, while **infrared sensors** have **limited range** and accuracy. Radar technology offers a revolutionary approach that overcomes these challenges.
- **Rising Need:** Modern security demands require proactive and intelligent systems for intrusion detection across sensitive areas.
- Conventional Limitations: CCTV:
  - Vulnerable to low-light, fog, heavy rain, and physical obstructions, leading to detection failures and false alarms.

Millimeter-wave radar offers superior performance in adverse environmental conditions, making it an ideal candidate for next-generation intrusion detection systems. Its ability to penetrate fog, smoke, and operate effectively in complete darkness provides a distinct advantage over optical systems.

The Imperative: Smarter Perimeter Surveillance

Radar systems provide several critical advantages: they operate effectively in complete darkness, penetrate through fog and rain, offer precise distance measurements, and detect motion patterns that other sensors miss. The integration of data analytics transforms raw radar signals into intelligent security insights, enabling automated threat assessment and response.

Weather Resilient

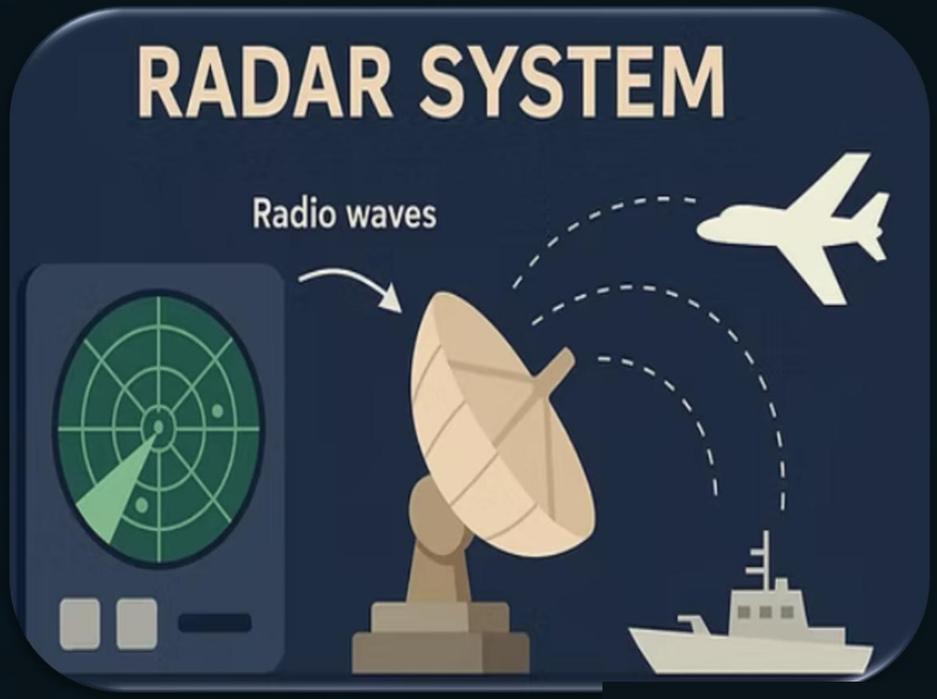
Operates effectively in rain, fog, dust, and darkness.

Privacy Preserving

Captures point-cloud data (dots representing objects) instead of identifiable images.

Rich Data

Provides not just location (X, Y, Z) but also velocity (Doppler) and reflectivity (intensity) of objects.



Project Overview: Building an Intelligent Radar System

Our project focuses on developing a sophisticated radar-based intrusion detection system that can not only detect but also accurately classify various types of intruders using advanced machine learning techniques and publicly available datasets.

Detect & Classify

The core objective is to identify and categorize intrusions, distinguishing between:

- Humans (e.g., intruders, personnel)
- Vehicles (e.g., cars, drones)
- Animals (e.g., wildlife, pets)

This granular classification enhances security response and reduces false alarms.

Leveraging Public Data & ML

We utilize rich, publicly accessible mmWave radar datasets to train and validate our models. This approach ensures reproducibility and broad applicability.

- Supervised machine learning for robust classification.
- Exploiting vast radar point cloud information.

Real-Time Monitoring Target

The ultimate goal is a deployable, real-time monitoring system tailored for critical applications:

- Defense installations
- Critical infrastructure protection
- Border surveillance

Ensuring immediate and accurate threat assessment.

Project Roadmap: From Raw Data to Deployment

Our development process follows a structured, iterative approach, ensuring thoroughness at each stage, from initial data exploration to the final deployment of the intrusion detection system.

1 **Download & Explore Dataset**

Acquire raw radar data and perform initial sanity checks.

2 **Label Data**

Annotate raw radar point clouds with corresponding object classes.

3 **Feature Extraction**

Derive meaningful features from the raw radar measurements.

4 **Preprocessing**

Clean, normalize, and prepare data for machine learning models.

5 **Model Training**

Develop and optimize machine learning models for classification.

6 **Results Visualization**

Present model performance and detection capabilities.

7 **Dashboard Design**

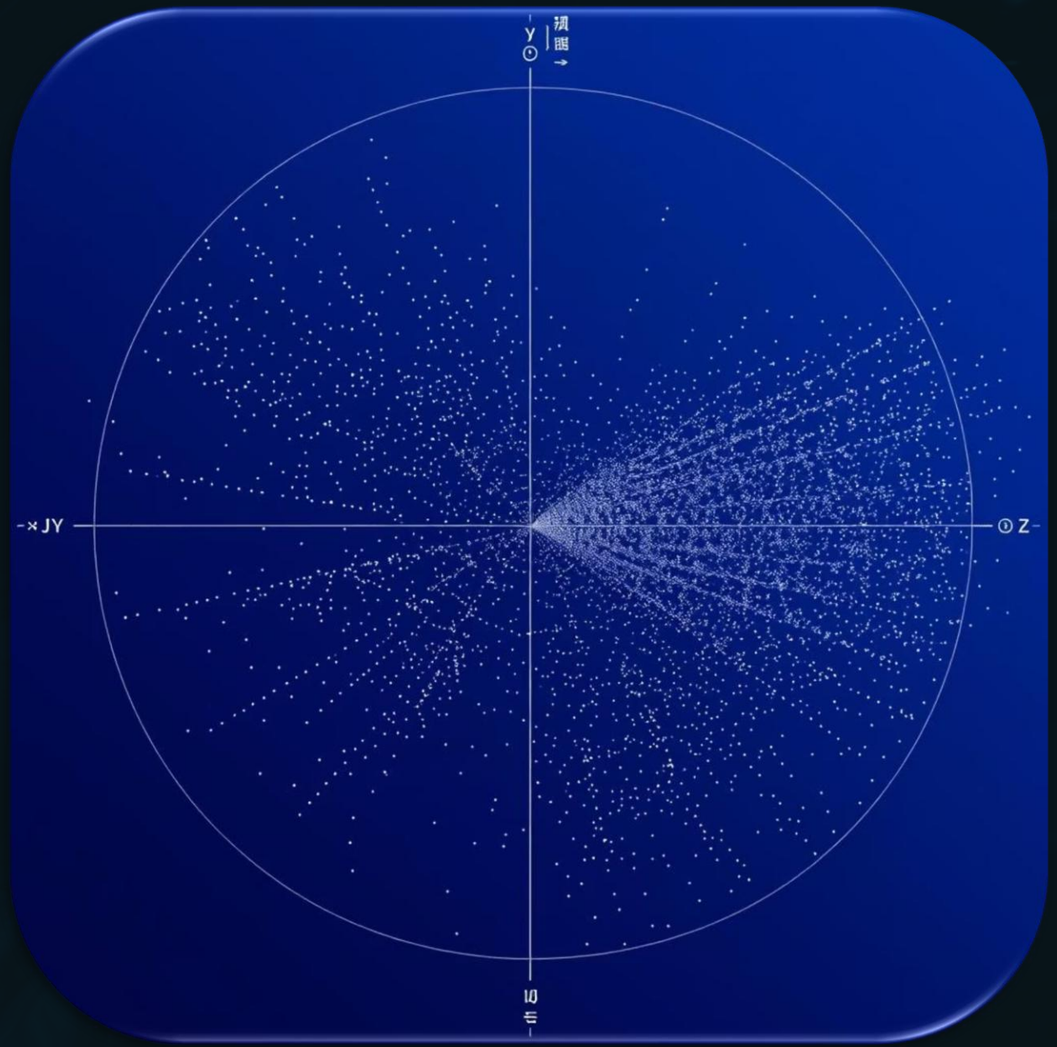
Create an intuitive interface for real-time monitoring.

8 **Deployment**

Integrate and test the system in a target environment.

Dataset Deep Dive: Open Urban mmWave Radar & Exploration

For our project, we are primarily leveraging the comprehensive Open Urban mmWave Radar Dataset from Zenodo. This dataset is specifically designed for urban sensing and classification tasks, providing a rich source of real-world radar signatures.



Key Characteristics:

- Diverse Scenarios: Captures various objects and scenarios relevant to perimeter security, including:
  - Humans (walking, running, standing)
  - Stationary objects (walls, poles, furniture)
  - Common items (backpacks, bags)
- Raw Data Format: Provided as CSV files, making it accessible for direct processing and feature extraction.
- Rich Point Cloud Data: Each entry in the dataset represents a Radar detection point with detailed attributes.

Data Labeling for Supervised Learning

Accurate data labeling is a critical prerequisite for training robust supervised machine learning models. This step transforms raw radar detections into meaningful, categorized instances.

Object Categorization

Each cluster of radar points corresponding to an object needs to be assigned a specific class label. Our primary categories include:

- **Human:** Individuals moving within the radar's field of view.
- **Wall:** Static environmental structures that reflect radar signals.
- **Backpack:** Smaller objects, potentially carried by humans, presenting distinct signatures.
- Other relevant objects defined within the dataset's metadata.

Labeling Methodologies

Two main approaches are considered for assigning these labels:

- **Metadata-Based Labeling:** Utilizing existing annotations or scene descriptions provided with the Zenodo dataset, which often includes ground truth information. This is the preferred method when available.
- **Manual Annotation:** For ambiguous or un-labeled sequences, expert human annotators review the point cloud data and assign labels. This can involve custom tools for visualizing 3D point clouds and marking regions of interest.

Feature Extraction from Radar Point Clouds

Raw radar point clouds are rich in information, but direct use can be computationally intensive and noisy. Feature extraction condenses this raw data into a more concise and descriptive representation, highlighting characteristics relevant for classification.

Clustering Techniques

The first step often involves grouping individual radar points into meaningful clusters, each potentially representing a distinct object.

- **DBSCAN (Density-Based Spatial Clustering of Applications with Noise):** Effective for identifying clusters of varying shapes and handling noise, ideal for irregular radar point distributions.
- **K-Means:** Can be used for initial grouping, though it assumes spherical clusters.

Each identified cluster then forms the basis for extracting object-level features.

Key Features Extracted

From these point clouds and their clusters, we derive a set of discriminative features:

- **Range (Distance):** Distance of the object from the radar sensor, calculated from x,y,z coordinates.
- **Doppler Velocity:** The radial speed of the object towards or away from the sensor. This is crucial for distinguishing moving intruders from static clutter.
- **Reflectivity (Intensity):** The strength of the radar return, which can provide clues about the object's material composition and size.
- **Object Dimensions:** Derived from the spatial extent of the point cloud cluster (e.g., length, width, height, volume).
- **Point Density:** Number of points within a cluster, indicating object solidity or presence.

Data Preprocessing for Model Readiness

Once features are extracted, the data undergoes rigorous preprocessing to ensure quality, consistency, and optimal format for machine learning model training. This step directly impacts model performance and reliability.

Cleaning and Imputation	• **Handling Missing Values:** Strategies such as mean imputation, median imputation, or removal of incomplete entries, depending on the extent and nature of missingness. • **Outlier Detection & Removal:** Identifying and addressing anomalous data points that could skew model training, often using statistical methods (e.g., Z-score, IQR).
Resampling for Balance	• **Addressing Class Imbalance:** In intrusion detection, "no intrusion" events often heavily outweigh "intrusion" events. Techniques like oversampling (e.g., SMOTE) or undersampling are employed to create balanced classes, preventing models from being biased towards the majority class.
Normalization/Standardization	• **Feature Scaling:** Applying techniques like **StandardScaler** to transform features to a common scale (e.g., zero mean and unit variance). This is critical for distance-based algorithms and gradient-descent optimization, ensuring all features contribute equally to the model.
Data Persistence	• **Saving Preprocessed Data:** Storing the cleaned and transformed data in an optimized format (e.g., Parquet, HDF5, or serialized Python objects) for efficient loading during model training and evaluation, saving significant time in iterative development.

Training the Classifiers

Once features are extracted, the data undergoes rigorous preprocessing to ensure quality, consistency, and optimal format for machine learning model training. This step directly impacts model performance and reliability.

Models Explored:

🌿 **Random Forest**

- Ensemble-based model combining multiple decision trees
- Robust to noise and high-dimensional radar features
- Primary model due to strong performance and low overfitting

🌳 **Decision Tree**

Simple and interpretable

Helped visualize decision boundaries and understand feature importance

Used for exploratory analysis and insight generation

Future Exploration:

🧠 **CNNs (Convolutional Neural Networks):**

- Can process radar-generated 2D/3D heatmaps
- Useful for spatial-temporal feature extraction

🌀 **PointNet:**

- Deep learning architecture for raw point cloud data
- Learns directly from unordered radar point sets without preprocessing

Random Forest

- **Why Random Forest?**
  - It builds multiple decision trees and merges their results, leading to higher accuracy and stability.
  - It can effectively capture the non-linear relationships between our extracted features.
- **Performance:** Our trained Random Forest model achieved high accuracy in classifying objects.
- **Example Result:** The model demonstrated a ***100%*** accuracy on the test set, proving its effectiveness.

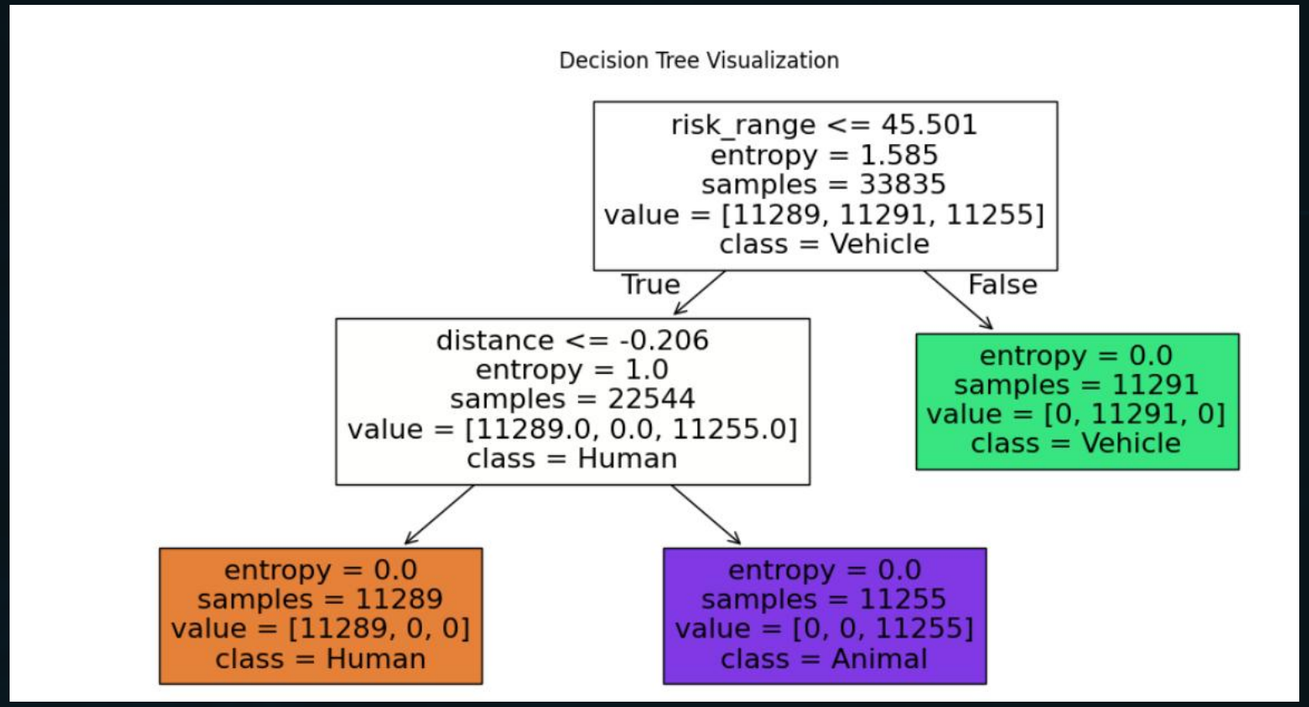
```
Python 3.13.1 (tags/v3.13.1:0671451, Dec  3 2024, 19:06:28) [MSC v.1942 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.

= RESTART: C:/Users/Admin/Desktop/Intrusion-detection-project/Intrusion-Detection-Project/notebooks/random forest acc.py
Random Forest Accuracy: 1.0

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      2809
           1       1.00      1.00      1.00      2807
           2       1.00      1.00      1.00      2843

    accuracy                           1.00      8459
   macro avg       1.00      1.00      1.00      8459
weighted avg       1.00      1.00      1.00      8459
```

Decision Tree

Why Decision Tree ?

- Ensemble of multiple decision trees, aggregating their outputs
- **Advantages:**
  - Higher accuracy and generalization
  - Handles non-linear feature relationships better

```
Dataset shape: (42294, 12)
Label distribution:
label
1    14098
2    14098
0    14098
Name: count, dtype: int64

Training set size: 33835
Test set size: 8459

==================================================
Training Decision Tree Classifier
==================================================
Decision Tree Accuracy: 1.0

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      2809
           1       1.00      1.00      1.00      2807
           2       1.00      1.00      1.00      2843

    accuracy                           1.00      8459
   macro avg       1.00      1.00      1.00      8459
weighted avg       1.00      1.00      1.00      8459
```



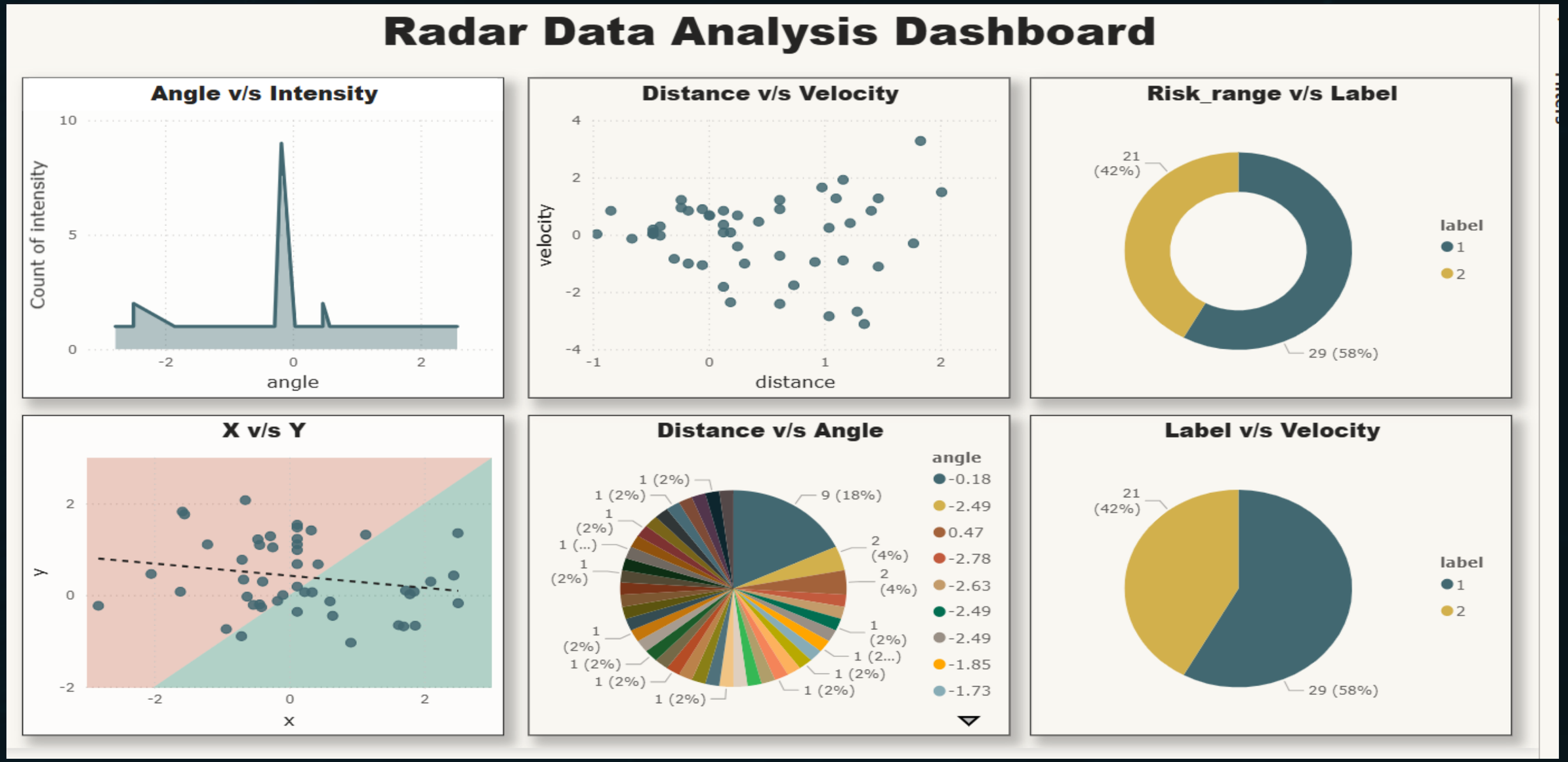
Sample prediction of Models

Sample prediction refers to using a trained machine learning model to predict the **class label** of **new, unseen data** (e.g., whether a radar point cloud belongs to a human, vehicle, wall, etc.). It helps **evaluate** how well the model generalizes beyond the training set.

```
Sample predictions:
Features (first 5 test samples):
[[-0.22278548  0.37169869 -1.85146577 62.74905797]
 [ 0.08101818 -0.47962025 -0.28587979 49.25319666]
 [ 0.99242918 -0.3580032   1.49403109 48.25106036]
 [ 0.84052733  0.18927348 -1.73126248 54.36064737]
 [ 0.90550528  1.2230181   0.14319041 32.7669013 ]]
Decision Tree predictions: [1 1 1 1 2]
Random Forest predictions: [1 1 1 1 2]
Actual labels: [1 1 1 1 2]
Decision tree visualization saved as 'decision_tree_visualization.png'

==================================================
Summary
==================================================
Decision Tree Accuracy: 1.0000
Random Forest Accuracy: 1.0000

Files created:
- decision_tree_model.pt
- random_forest_model.pt
- model_metadata.pt
- decision_tree_model.pkl
- random_forest_model.pkl
- decision_tree_visualization.png (if matplotlib works)
```

```
==================================================
Testing Model Loading
==================================================
Successfully loaded all models!
Decision Tree type: DecisionTreeClassifier
Random Forest type: DecisionTreeClassifier
```

The Dashboard



Conclusion & Future Work

Achievements:

- Successfully built an end-to-end pipeline for radar-based intrusion detection.
- Demonstrated high accuracy using a Random Forest classifier on a public dataset.
- Designed a functional prototype for a real-time visualization dashboard.

Future Work:

- Explore More Datasets: Incorporate different datasets (like CARRADA) to improvemodel generalization.
- Advanced Models: Implement and evaluate deep learning models like PointNet forpotentially higher accuracy.
- Real-World Testing: Deploy the system with a live mmWave radar sensor (e.g., a TI IWRseries) on an edge device.