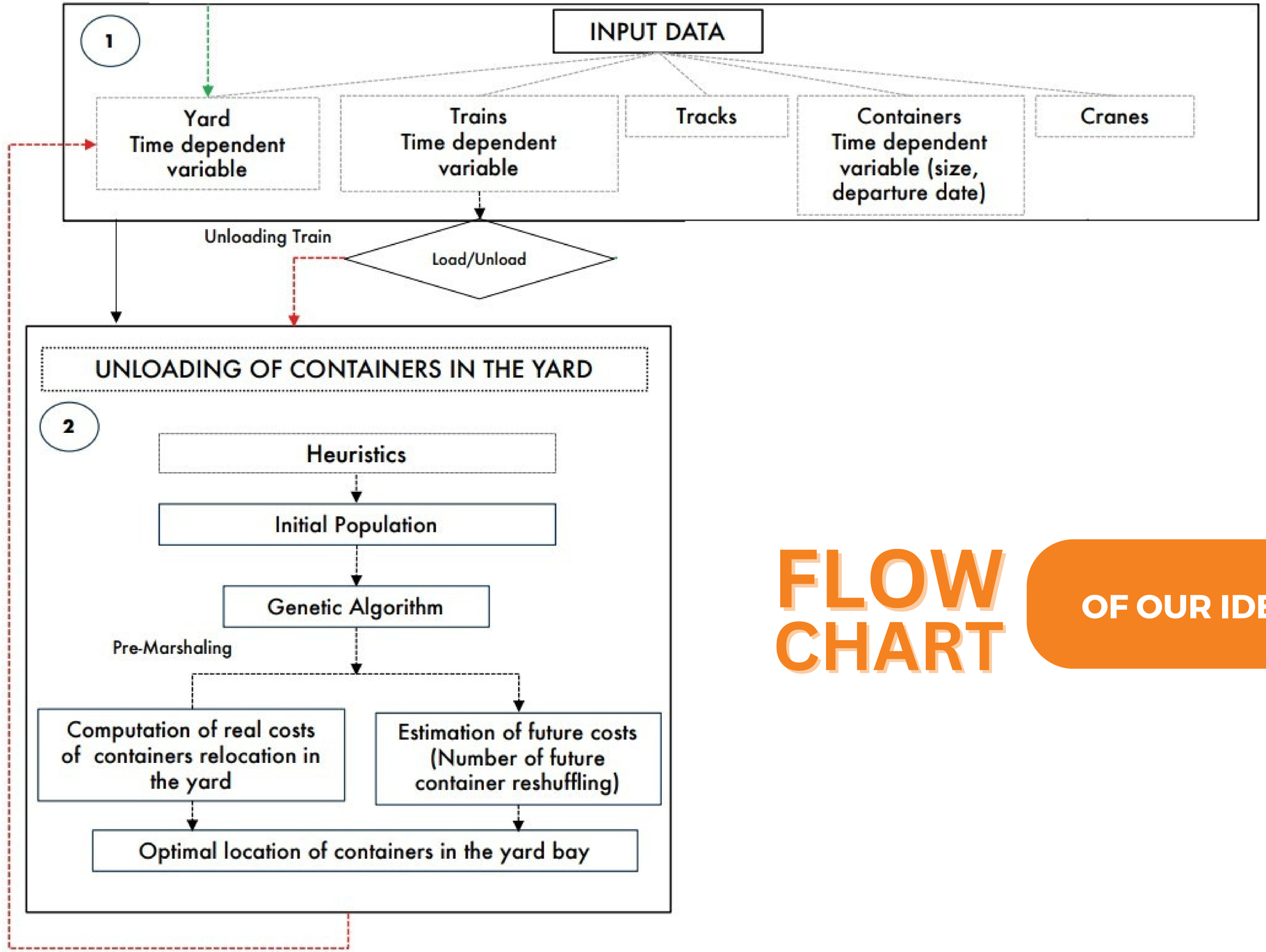

EFFICIENT
CONTAINER
PLACEMENT
SYSTEM
EFFIC
YARD MANAGEMENT SYSTEM

PROBLEM STATEMENT

The goal is to make a computer system that uses AI to place containers in the best way. It needs to be fast and put containers in the right spots. It should also be able to guess when containers will leave and where they should go. It also has to be safe with people's information and used in a good way.

BASICALLY,

We're designing a user-friendly platform where users can enter container information and when it arrived in the yard. The system will then provide the optimal spot for the container to be placed, aiming to reduce the need to move other containers around.



FLOW CHART

OF OUR IDEATION

SOLUTION TASKS

1. We intend to employ machine learning algorithms for predicting the OUT_TIME of individual containers.

2. We will use the graph neural network model to find the optimal position of the container.

3. We will combine the output of Time series algorithms and graph search Algorithms to minimize shuffles.

SOLUTION TASK 1

PREDICTING THE OUTGOING DAYS OF INCOMING CONTAINERS BASED ON PAST DATA

1.1 Data Integration and Refinement:

Through the grouping of variables including arrival timestamps, container dimensions, and status indicators, we precisely prepared a comprehensive dataset. Subsequently, this refined dataset found application within a Random Forest Regression Model: a sophisticated algorithmic construct renowned for its multi-tree learning capacity.

SOLUTION
TASK 1

PREDICTING THE OUTGOING DAYS OF INCOMING CONTAINERS BASED ON PAST DATA

1.2 Synergistic Predictive Framework

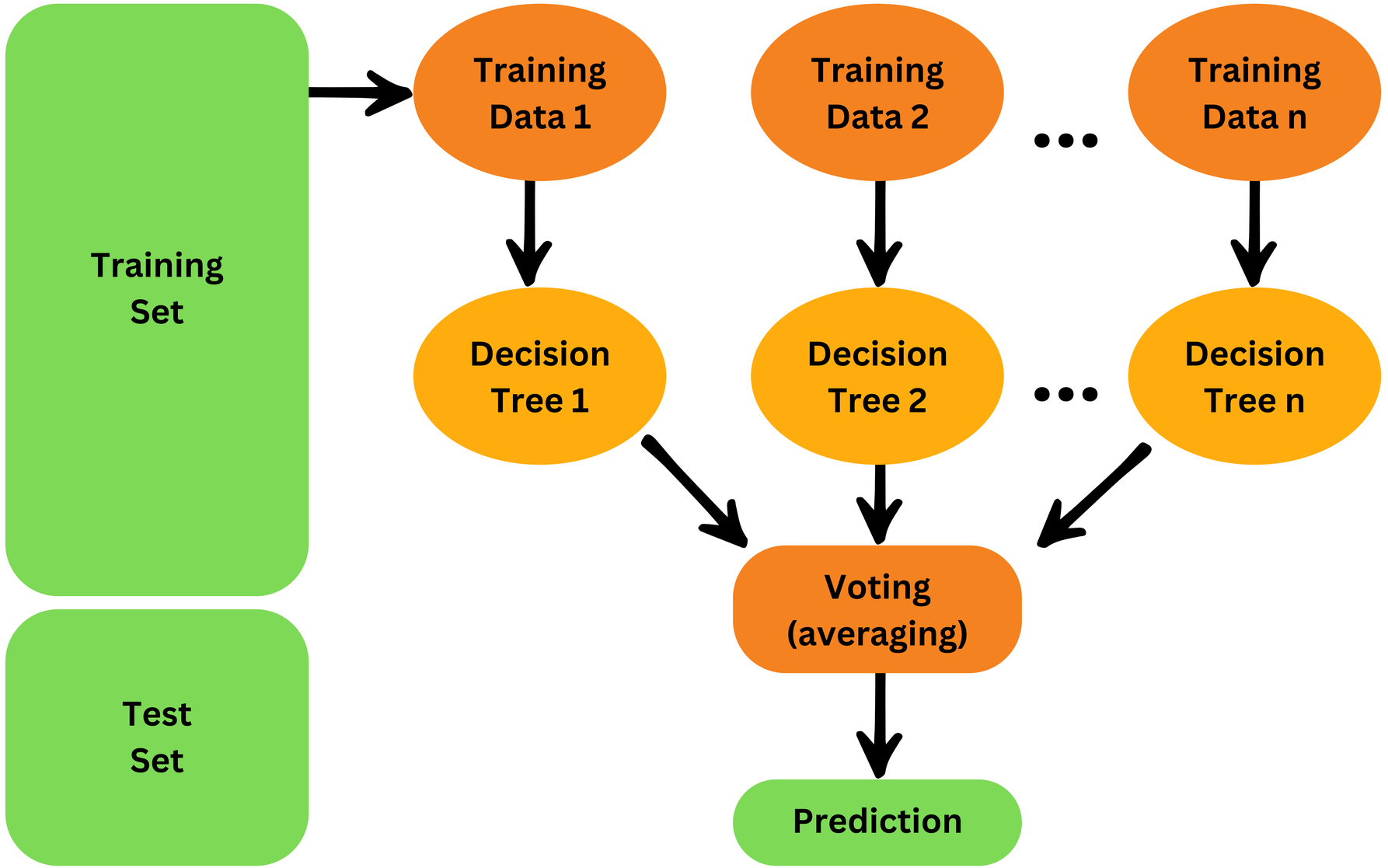
In this model, a group of individual regression decision trees worked together, similar to a panel of different subject matter experts. Their combined predictions were coordinated to create a comprehensive projection, thus utilizing the synergy of their collective knowledge.

SOLUTION TASK 1

PREDICTING THE OUTGOING DAYS OF INCOMING CONTAINERS BASED ON PAST DATA

1.3 Methodical Application

Using the model required providing vital inputs like arrival times, container sizes, and operational statuses. This produced an expected departure date (OUT_DATE) determined by a balanced combination of computational expertise and data-driven insight.


Training Set
Test Set
Training Data 1
Training Data 2
Training Data n
Decision Tree 1
Decision Tree 2
Decision Tree n
Voting (averaging)
Prediction

Output of 'Predicted leave date of the container'

```
import pandas as pd
import warnings

warnings.filterwarnings("ignore")

model = pickle.load(open('Out Date Prediction model','rb'))

in_date = input("\nEnter In Date & Time (dd-mm-yy hh:mm:ss):")
con_size = int(input("Enter Container Size:"))
status = input("Enter Status (E/L):")


if status.upper()=="L":
    status=1
else:
    status=0

in_date = dt.datetime.strptime(in_date, '%d-%m-%y %H:%M:%S')
```

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS D:\DAIICT>  & 'C:\Users\Darsh K Thakkar\AppData\Local\Program\Python\Python311\python.exe' 'c:\Users\Darsh K Thakkar\.vscode\extensions\ms-python.python-2023.14.0\pythonFiles\lib\python\debugpy\adapter/../..\debugpy\launcher' '49731' '-' 'D:\DAIICT\Predict Date.py'

Enter In Date & Time (dd-mm-yy hh:mm:ss):24-11-22 18:42:15
Enter Container Size:40
Enter Status (E/L):E

The expected OUT_DATE of the container: 2022-11-25
PS D:\DAIICT>
```

1 Techniques used for Predicting the OUT_TIME of the container.

- Data Cleaning
- Data Preprocessing using an Ordinal Encoder
- Training the Random Forest Regression Model

SOLUTION TASK 2

PREDICTING OPTIMAL LOCATION FOR NEW CONTAINER IN YARD

2.1 Graph Neural Networks (GNNs) are a class of machine learning models specifically designed to work with graph-structured data. In our case, using GNNs is an effective way to predict the optimal location for a new container in a yard where containers and their relationships are represented as a graph.

SOLUTION TASK 2

PREDICTING OPTIMAL LOCATION FOR NEW CONTAINER IN YARD

2.2 Yard is represented as a graph. Each container is a node in the graph, and the relationships between containers (e.g., based on 'Area', 'Bay', etc.) are represented as edges connecting these nodes.

2.3 The core of a GNN is the neural network architecture that operates on the graph data. A GNN has message-passing layers that allow nodes to aggregate information from their neighbors. In this case, GraphConv layer provided by DGL library is used.

SOLUTION TASK 2

PREDICTING OPTIMAL LOCATION FOR NEW CONTAINER IN YARD

2.4 GNN is trained to learn representations of the containers in the yard. labeled data (occupied or not) is used to guide the training process. The GNN learns to capture the spatial dependencies and relationships between containers in the yard.

SOLUTION TASK 2

PREDICTING OPTIMAL LOCATION FOR NEW CONTAINER IN YARD

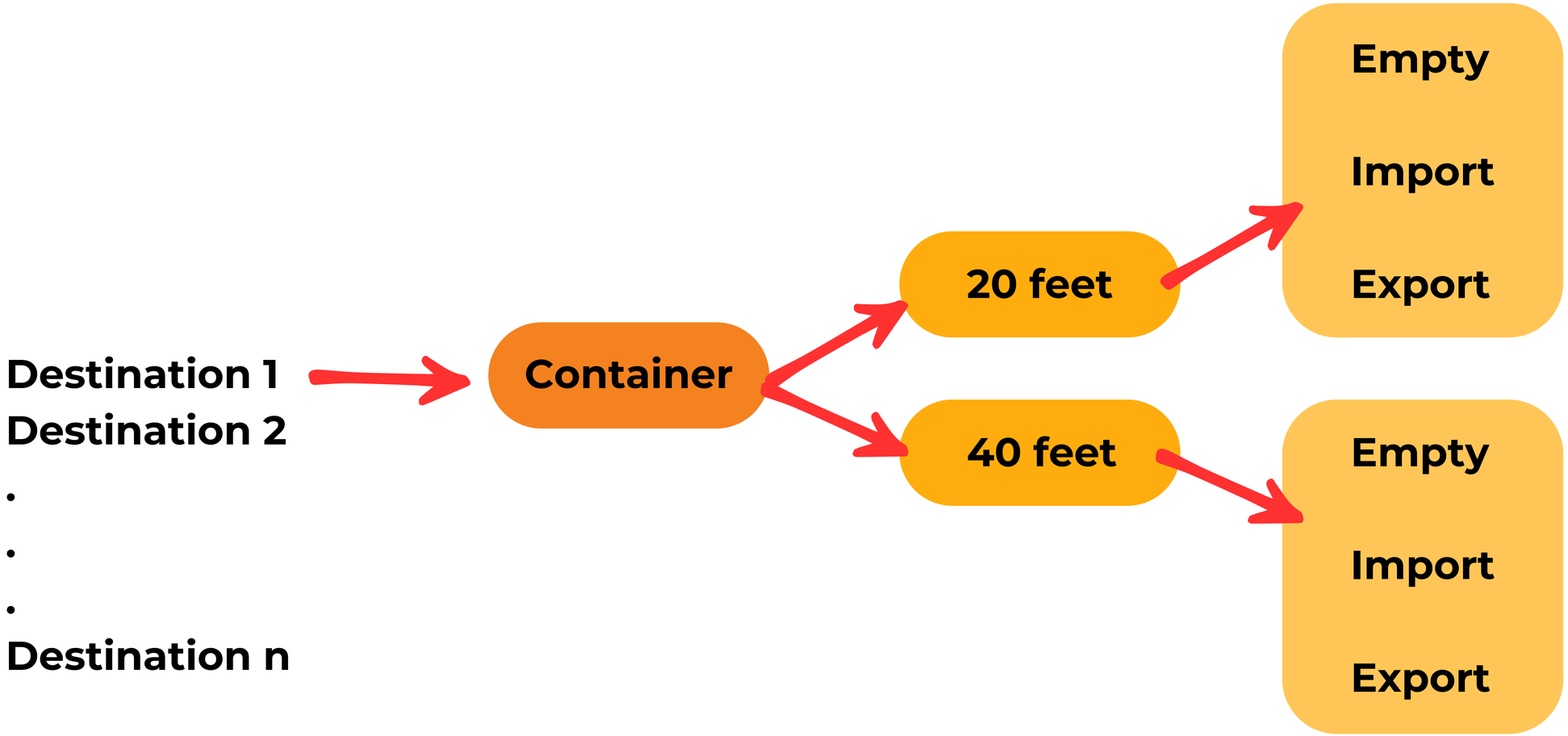
2.5 To predict the optimal location for a new container, we will feed its features into the trained GNN. The GNN will propagate information through the graph, taking into account the relationships and dependencies between the containers. The final layer's output can be used to predict whether the location will be occupied or not.

Output to find the optimal position of the container.
(when new container is arrived)

```
# Data preprocessing
container_data['IN TIME'] = pd.to_datetime(container_data['IN TIME'])
container_data['OUT TIME'] = pd.to_datetime(container_data['OUT TIME'], errors='coerce')

merged_data = pd.merge(container_data, yard_data, how='left', left_on=['CON_SIZE', 'CON_NUM'], right_on=['Container Size', 'Location'])
merged_data['Stay Duration'] = (datetime.now() - merged_data['IN TIME']).dt.total_seconds()

# Feature Engineering
label_encoders = {}

categorical_features = ['Area', 'Bay', 'Location Status', 'STATUS']
for feature in categorical_features:
    le = LabelEncoder()
    merged_data[feature + '_encoded'] = le.fit_transform(merged_data[feature])
    label_encoders[feature] = le

features = ['Area_encoded', 'Row', 'Bay_encoded', 'Level', 'CON_SIZE', 'STATUS_encoded', 'Stay Duration']

# Create graph
```

```
C:\Users\NISHIT\PycharmProjects\pythonProject111\venv\Scripts\python.exe C:/Users/NISHIT/PycharmProjects/pythonProject111/main.py
Validation Accuracy: 0.8125
Optimal location for the new container: A02A3

Process finished with exit code 0
```

To Determine block of the container (Laded / Empty)



2 Techniques used for finding the optimal position of the container.

- GNN
- DGL library
- PyTorch Tensors
- GraphConv provided by DGL
- Loss function: CrossEntropy
- Adam Optimizer

SOLUTION

TASK 3 TO MINIMIZE SHUFFLES

3.1 Timeseries Algorithms:

Timeseries algorithms are used to analyze data that changes over time, like the historical records of container arrivals and departures. These algorithms can predict future values based on past patterns and trends. It can be used to predict when a container is likely to leave the yard. This prediction helps in planning where to place the container optimally so that it's easily accessible when it's time to remove it.

SOLUTION

TASK 3 TO MINIMIZE SHUFFLES

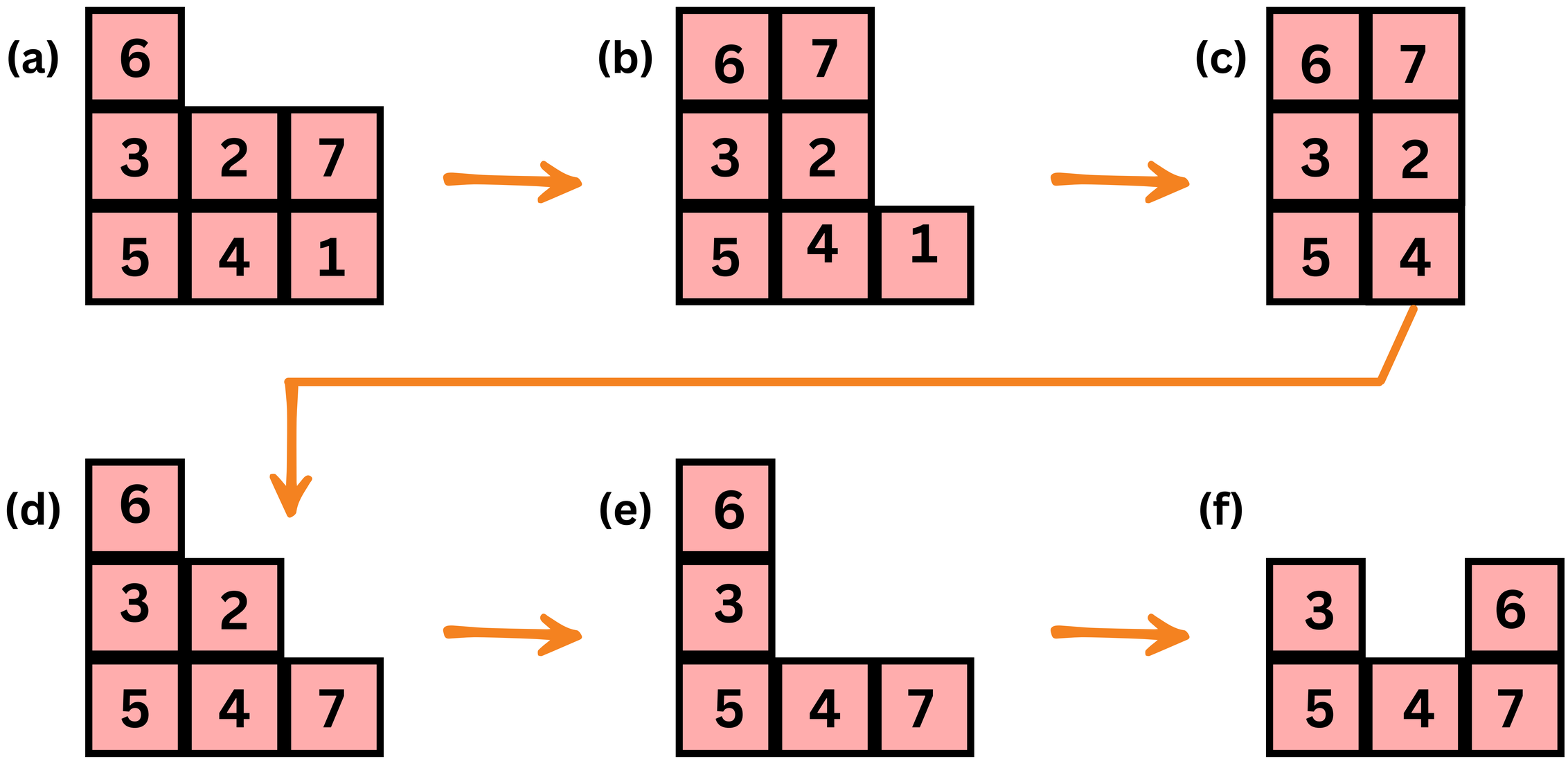
3.2 Graph Search Algorithm:

A graph search algorithm is a technique to traverse a graph (a network of interconnected nodes) to find a specific solution. The yard layout can be represented as a graph where containers are nodes, and edges represent their spatial relationships. By using a graph search algorithm, you can determine the best position to place a new container that minimizes the need to move other containers when it need to be removed. This ensures that the yard layout remains as undisturbed as possible, reducing unnecessary shuffling.

SOLUTION TASK 3 TO MINIMIZE SHUFFLES

3.3 Combining the Approaches:

The combination of these two approaches involves using the prediction from the timeseries algorithm (when a container is expected to leave) and integrating it with the graph search algorithm to decide where to place the container. By considering the anticipated departure time and the existing spatial relationships between containers, the system can strategically position the container to minimize shuffling during removal.


(a)
6
3 2 7
5 4 1
(b)
6 7
3 2
5 4 1
(c)
6 7
3 2
5 4
(d)
6
3 2
5 4 7
(e)
6
3
5 4 7
(f)
3 6
5 4 7

3 Techniques used for Reshuffling

- **Mathematical model calculation:**
  - Integer Mathematical Model
- **Reshuffling of Container**
  - Hybrid Meta Heuristic Approach
- **Greedy Algorithm Approach**
- **For Optimization:**
  - Beam Search Algorithm

FUTURE WORK:

1

Augmented Reality (AR) Visualization:

Develop an AR interface that overlays digital information onto the physical yard layout. Yard managers could use AR glasses or devices to see real-time container locations, optimal placement suggestions, and other relevant data.

FUTURE WORK:

2

Route Optimization for Yard Vehicles:

Implement a route optimization module within your system that calculates the most efficient paths for yard vehicles like forklifts and trucks. This feature would help minimize travel distances, reduce fuel consumption, and optimize the movement of assets within the yard.

FUTURE WORK:

2

1 Entry gate to yard Monitored by Arduino Portenta H7

Upon detection, ML model is run

2 Detected license plate is shown as QR code on screen at gate

3 Driver scans it with our AppGyver

Webservice: AppGyver retrieves Yard Order from Yard Logistics

4 Driver gets overview of all tasks; Proceeds to destination

5 Upon arrival, task is executed & driver scans a static QR that's placed there

6 Task in Yard Logistics is confirmed

Driver continues with subsequent tasks Process repeats until all done

7

Truck leaves yard

FUTURE WORK:

3

Real-time Container Tracking:

Enhance the visibility of your system by integrating real-time tracking technologies like RFID or GPS for containers. This would provide accurate location information, allowing for more precise container placement decisions.

The UI of the Yard Management Software is attached below:

id=0%3A1&mode=design&t=3es0fJqFEdlim1oz-1