This paper aims to provide a description of the progress of the code developed for the training of a machine learning model for the creation of an autonomous navigation system for a four-wheeled omnidirectional robot. This progress is focused on the explanation of the first phase of the final project.

Initial Supervised Learning: In this initial stage, a dataset called "Initial Training Dataset" containing labeled information about sensors and control actions will be used. This dataset will be obtained as the project progresses by researching and collecting data from different sources, including:

- Online Resource Exploration: extensive research will be conducted in online resources, such as Kaggle, the UCI Machine Learning Repository, and other relevant sources to search for public datasets related to robotics and autonomous navigation.

- Progressive Collection: As suitable datasets are identified; we will proceed with progressive collection of the datasets. This may include downloading publicly available datasets or collecting additional data through testing and experimentation in the controlled environment of the project.

- Labeling and Preparation: All data, whether from public sources or from the custom dataset, will be labeled and prepared appropriately for use in the supervised learning model training process. This may require the identification of correct and incorrect actions in the data, as well as the creation of appropriate annotations.

The dataset it was decided to implement is the IRND found in Kaggle.

This data was collected for performing Autonomous Navigation in an indoor environment. The dataset comprises of data gathered by navigating a real robot over 2 kinds of surfaces.

Content

The entire dataset has 276 files in total and has been organised into 2 folders, each for storing the data collected from one surface. outputs_2 folder contains data collected from a smooth surface and outputs folder contains data collected from a rough surface.

Each file of the dataset is a json file that stores a sequence data recorded from 1 episode of robotic control, i.e, data collected from robotic sensors and actuators while controlling the robot from an initial position at rest to the target position. The data collected at each time step of an episode contains the following records:

- num_records: no. of records in an episode
- direction: clock-wise/counter-clockwise depending upon the direction of movement of robot
- pose: current location/position of robot
- brake: 1 if brake is applied, 0 otherwise
- angles: obtained from LiDAR. Ranges from -180 degrees to +180 degrees

- dists: obstacle distances obtained from LiDAR corresponding to respective LiDAR angles
- horn: 1 if pressed, otherwise 0
- counts_left: speed of left wheel. Max speed = 2000 counts
- counts_right: speed of right wheel. Max speed = 2000 counts

Experimentally it was found that approximately 800 counts corresponds to 1 ft/s speed

A sample model for this dataset to predict the surface on which the robot is controlled, has been provided for reference.

The IRND was selected for the following key reasons:

1.Controlled Environment: Collected in an indoor environment, it provides consistent data.

2. Lidar Sensor: Includes crucial lidar data for accurate sensing of the environment.

3. Sensors and Actuators: Provides a variety of information, including wheel speed.

4.Project Relevance: Designed for the development of autonomous navigation systems, aligning with the specific objectives of the current project.

The code starts with the implementation of the following libraries.

```
import os
import json
import zipfile
import matplotlib.pyplot as plt
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
```

- os:

Description: This library provides an interface to the underlying operating system. It is used to manipulate file and directory paths, as well as to perform file system related operations.

- json:

o Description: The json library is used to work with data in JSON format. In this case, it is used to read JSON files containing data about the robot's navigation.

- zipfile:

o Description: The zipfile library allows working with files compressed in zip format. It is used to extract the contents of the zip file containing the IRND dataset.

- matplotlib.pyplot:

o Description: matplotlib.pyplot is a visualization library used to create graphs and visualizations. In the code, it is used to print information about the number of files in the outputs and outputs_2 directories.

- numpy:

o Description: numpy is a fundamental library for performing numerical operations in Python. It is used to work with arrays and matrix operations. In this code, it is used to manipulate the characteristics (distances) of the data set.

- sklearn.model_selection.train_test_split:

o Description: from the scikit-learn library, train_test_split is used to split the dataset into training and test sets. It allows to create separate datasets to train and test the model.

- sklearn.tree.DecisionTreeClassifier:

o Description: decisionTreeClassifier is an implementation of decision trees in scikit-learn. It is used to create and train a decision tree model in order to predict the surface on which the robot is located.

In this part of the code, it is used to be able to detect the IRND dataset and decompress it for use.

```
zip_file_path = "/content/IRND.zip"
extracted_dir = "/content/IRND"

with zipfile.ZipFile(zip_file_path, 'r') as zip_ref:
    zip_ref.extractall(extracted_dir)
```

The path to the directory containing the outputs and outputs_2 files is specified. Then, all JSON files present in these directories are read, storing the corresponding information in the data_outputs_1 and data_outputs_2 lists.

```
# Ruta al directorio que contiene los archivos outputs y outputs_2
data_dir = "/content/IRND"

# Obtener todos los archivos JSON en los directorios
json_files_1 = [os.path.join(data_dir, 'outputs', f) for f in
os.listdir(os.path.join(data_dir, 'outputs')) if f.endswith('.json')]
```

```
json_files_2 = [os.path.join(data_dir, 'outputs_2', f) for f in
os.listdir(os.path.join(data_dir, 'outputs_2')) if
f.endswith('.json')]

# Leer todos los archivos JSON en los directorios
data_outputs_1 = []
for file in json_files_1:
    with open(file, 'r') as f:
        data_outputs_1.append(json.load(f))

data_outputs_2 = []
for file in json_files_2:
    with open(file, 'r') as f:
        data_outputs_2.append(json.load(f))
```

The features (X) are extracted from the data_outputs_1 data set. In this case, the distances (dists) obtained from the sensors are used as model features. The data is then split into training and test sets using the train_test_split function of scikit-learn.

```
print("Número de archivos en outputs:", len(data_outputs_1))
print("Número de archivos en outputs_2:", len(data_outputs_2))
# Características (X): Usar distancias como características
X = [item['dists'] for data_output in data_outputs_1 for item in
data_output['data']]

# División en Conjuntos de Entrenamiento y Prueba
X_train, X_test, y_train_encoded, y_test_encoded = train_test_split(X,
y_encoded, test_size=0.2, random_state=42)
```

In Section 2, labels are encoded using the LabelEncoder class of scikit-learn. The labels are obtained from the all_data dataset by creating the y-list. These encoded labels are used in the division of training and test sets.

```
# Sección 2: Codificación de Etiquetas con LabelEncoder
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

# Características (X): Usar distancias como características
X = [item['dists'] for data_output in data_outputs_1 for item in
data_output['data']]

# Codificación de etiquetas
y_encoded = le.fit_transform(y)
```

```
# División en Conjuntos de Entrenamiento y Prueba
X_train, X_test, y_train_encoded, y_test_encoded = train_test_split(X,
y_encoded, test_size=0.2, random_state=42)
```

Dataset Selection (IRND):

In the search for the right dataset for the project, several crucial aspects were considered. The IRND stood out as the ideal choice for the following reasons:

- Controlled Environment: The IRND was compiled specifically for autonomous navigation in indoor environments. This feature provides consistent and beneficial data for initial prototype development.
- Lidar Sensor: Includes detailed data from the lidar sensor, essential for accurate sensing of the environment. Angles and distances obtained from the Lidar are critical for decision making in autonomous navigation.
- Sensor and Actuator Data: IRND provides a variety of data, from sensors such as Lidar to actuator information such as wheel speed. This data is valuable for training models that connect the perception of the environment with the control actions of the robot.
- Project Relevance: The dataset was created for the specific purpose of developing autonomous navigation systems, aligning closely with the objectives of our project.

Algorithm Selection (Decision Tree):

The choice of the decision tree algorithm was based on several considerations. It was considered feasible for the following reasons:

- Clear Interpretation: Decision trees provide a clear and simple interpretation of the model's decisions, facilitating the understanding of its operation.
- Mixed Data Handling: It can effectively handle mixed data, such as categorical and numerical variables, present in our data set.
- Scalability: For initial projects, a decision tree is scalable and does not require complex adjustments, making it suitable for initial prototype development.
- Good Generalization: With the ability to generalize well across different datasets, decision trees can adapt to the complexities of our autonomous navigation problem in indoor environments.