Mohammad Rohaan 22I-2327 ML For Robo A# 04

Abstract:

We worked on a machine learning project to help robots identify objects. Our goal was to create a system that can classify objects based on what sensors on the robot detect like how away something is, its size, color, weight and voltage readings. We used two methods to classify objects: K-Nearest Neighbors (KNN) and Naïve Bayes. We. Tested these methods using fake data that simulates what robotic sensors would see.

We checked how well both methods worked using metrics like accuracy, precision, recall, F1-score and confusion matrices. The KNN method was 31% accurate. The Naïve Bayes method was also 31% accurate.. Naïve Bayes was a bit better at picking out certain classes.

Our results show that both KNN and Naïve Bayes can be used for object classification in robots.. We need to work on a few things before they can be used in real-time like dealing with limited computer resources and noisy sensor readings. We think future work should focus on making these methods faster and more accurate, for object classification.

Introduction:

In robotic systems the ability to see and understand objects is really important for making good decisions especially in places like warehouses, factories or even self driving cars. Object classification helps robots work with their surroundings making choices based on the objects they see. The hard part is making systems that can correctly identify and group objects using information from sensors, such as how away something is, how big it is, how bright it is and sensor power.

This project is about creating a machine learning solution for object classification in a system using two well known methods: K-Nearest Neighbors and Naïve Bayes. The goal was to build a classification model that can identify different types of objects based on sensor information using a made up dataset that mimics real world sensor readings from sensors.

The importance of this work is that it can be used in robotic systems, where correctly identifying objects is critical for tasks like sorting picking up things and moving around. The models created in this project can be made better for real time classification, which is necessary in many robotic applications like object classification in robotic systems and object classification is really important, for this.

Problem Formulation:

The problem we are trying to solve is classifying objects for a system. This system uses sensor data to identify objects. The robot has to figure out what an object is based on what it senses, like how away it is, how big it is, what color it is, how heavy it is and what the sensor is telling it. The classification system has to put objects into groups that we already know about based on what the sensors tell it.

The object classification system is very important for the system. The robotic system needs to know what objects are so it can do its job. The object classification system uses sensor data to do this.

Input Features:

- Distance Readings: this is how far the robot is from the object. The robot can tell how away something is with a special sensor.
- Object Size: this is how big the object is. The robot can figure this out with cameras or special sound wave sensors.
- Color Intensity: this is how bright or what color the object is. The robot can tell this with color sensors or by looking at pictures of the object.
- Weight: this is how heavy the object is. The robot can figure this out with pressure sensors or scales.
- Sensor Voltage: this is what the sensors are telling the robot about the object. It is like information that helps the robot know what the object is.

Output Classes:

- Metal Can: the robot will learn to recognize this kind of object.
- Plastic Bottle: the robot will also learn to recognize this kind of object.
- Wooden Block: this is another kind of object the robot will learn about.
- Rubber Ball: this is a ball that the robot will learn to recognize.

Justification for Classification Approach:

We are using a kind of learning called supervised learning. This means we are teaching the robot with examples where we already know what the object is. The robot is going to use two ways to classify objects: K-Nearest Neighbors and Naïve Bayes.

K-Nearest Neighbors is a way of classifying objects based on what's around them. It looks at the objects that're most like the one it is trying to classify and uses those to make a decision. This works well for our problem because it can handle lots of kinds of data like how far away something is or how big it is.

Naïve Bayes is another way of classifying objects. It looks at how likely it's that an object is a certain thing, based on what we know about it. This works well when we do not have a lot of data or when we know what the data should look like.

Both of these methods are good for our problem because they can handle the kinds of data we are working with and they work well even when we do not have a lot of data. The object classification system uses these methods to classify objects, for the system. The robotic system uses the object classification system to know what objects are so it can do its job.

Data Preprocessing:

Data preprocessing is an important step when you are working with machine learning because it gets the raw data ready for the model to use. For this project we did a things to make sure the data was clean and ready to go.

Handling Missing Values:

The data we used for this project did not have any values.. If there were missing values we would have had to fill them in with something or get rid of the rows or columns that had missing data.

For example if we were missing a sensor reading like the distance we could fill it in with the distance from all the other readings.. We could just get rid of the rows that had missing data.

Feature Normalization:

Since we are using a model that is based on distance we need to make sure all the features are on the scale. This is because features with ranges like distance readings would be more important than features with smaller ranges if we did not do this.

We used something called StandardScaler to make all the features have the scale. This means that each feature will have an average of 0 and a standard deviation of 1.

We did this because our model uses something called distance to figure things out and this can be wrong if all the features are not on the same scale.

We used the following code to make the features all have the scale: scaler = StandardScaler() and data_scaled = scaler.fit_transform(data).

Encoding Categorical Data:

All the features in our data are numbers, like distance or size or weight so we did not have to do anything with data.

If we had categorical data, like the type of object we would have had to turn it into numbers that the model could understand.

For example if we had a type of object that was a string like "metal_can" we could turn it into a number, like 0 for metal cans or 1, for bottles.

Splitting the Dataset:

We split our data into two parts: a **training set** and a **test set**. This is so we could see how well the model would work on data it had not seen before.

We used the **training set** to teach the model. We used the test set to see how well the model could generalize or work with new data.

We used a function from scikit-learn to split the data. We made sure to use the same split every time by using a **test size** of 20% and a random state of 42.

Model Implementation:

This project used two machine learning algorithms for object classification: K-Nearest Neighbors and Naïve Bayes. Both models were implemented using the scikit-learn library because it is easy to use and has made functions for classification tasks.

The dataset was divided into input features. Target labels after preprocessing. The input features included distance, size, color intensity, weight and sensor voltage. The target label was the object class, which included metal can, bottle, wooden block and rubber ball classes.

K-Nearest Neighbors Implementation:

K-Nearest Neighbors is a classification algorithm that uses distance. The basic idea is that when the model gets an object it looks at the nearest points in the training data. Then it classifies the object based on the majority class.

In this project the K-Nearest Neighbors model was initially trained with n_neighbors = 3. This means the model looked at 3 neighbors for each new object. Later hyperparameter tuning was done to improve the performance of K-Nearest Neighbors. Different values of n_neighbors, weights and metric were tested using GridSearchCV.

The best K-Nearest Neighbors parameters were:

- metric = Euclidean
- n_neighbors = 7
- weights = distance

This means the final K-Nearest Neighbors model used Euclidean distance considered 7 nearest neighbors and gave more importance to closer neighbors.

Naïve Bayes Implementation:

Naïve Bayes is a classification algorithm that uses probability. It makes predictions based on probability. This model assumes that features are independent of each other. In real-world data this assumption is not always perfect. Naïve Bayes is still a fast and simple algorithm.

In this project Gaussian Naïve Bayes was used because the dataset features were numerical and continuous such as distance, size, weight and voltage. **GaussianNB** is suitable for data.

GridSearchCV was also used for the Naïve Bayes model to tune the var**_smoothing** parameter. The best value found was:

- **var_smoothing** = 1e-09

This parameter is used to give the model stability so that very small variance values do not cause calculation issues.

Model Training:

Both models were trained on the training dataset. The purpose of the training data was to teach the model the relationship between sensor readings and object classes. After training both algorithms were evaluated on the test dataset, which the model had not seen during training.

The implementation of K-Nearest Neighbors and Naïve Bayes was simple. Both algorithms worked differently. K-Nearest Neighbors used distance-based comparison while Naïve Bayes used probability-based classification.

Why These Models Were Selected?

K-Nearest Neighbors and Naïve Bayes were suitable for this project because they are both supervised classification algorithms. K-Nearest Neighbors is simple and intuitive. It can use distance patterns for sensor-based object classification. Naïve Bayes is fast, lightweight and useful in systems where computational resources are limited.

The main goal of this implementation was to train both algorithms on the dataset and compare them to see which model performed better for the robotic object classification problem.

Training and Validation:

So we made the models. Set their settings. Then we used the training data to teach them. Tested them on the test data. We trained the KNN model and the Naïve Bayes model. Checked how well they worked using some key measures.

Training the Models:

K-Nearest Neighbors (KNN):

We tried settings for the KNN model like how many neighbors to consider and how to measure the distance. Then we trained the KNN model using the training data.

We used a technique called cross-validation to make sure the model was good. This helped us find the combination of settings.

Naïve Bayes:

We did the thing for the Naïve Bayes model. We tried settings and trained the model using the training data.

We also used cross-validation to make sure the Naïve Bayes model worked well with data.

Evaluating the Models:

After we trained both models we tested them on the test data that they had not seen before. We used some measures to see how well they worked:

- Accuracy: This measures how correct predictions the model made. It is calculated by dividing the predictions by the total predictions.
- Confusion Matrix: This shows us how well the model worked for each class. It helps us see how times the model made correct predictions and how many times it made mistakes.
- Precision: This measures how many correct predictions the model made for a class. It is calculated by dividing the predictions by the total predictions for that class.
- Recall: This measures how correct predictions the model made for a particular class compared to all the instances of that class. It is calculated by dividing the predictions by the total instances of that class.
- F1-Score: This is a measure that combines precision and recall. It helps us see how well the model worked overall.

Model Evaluation Results:

KNN Model: The KNN model got 31% accuracy on the test data. The confusion matrix and classification report showed us how well the model worked for each class.

Naïve Bayes Model: The Naïve Bayes model also got 31% accuracy but it worked better for some classes. The classification report showed that it was better at predicting the metal_can class than the KNN model.

Performance Metrics:

- Accuracy: Both models got accuracy but other measures like precision and recall give us a better idea of how they worked.
- Confusion Matrix: The confusion matrix helped us see which classes the models struggled with.
- Precision, Recall and F1-Score: These measures give us a detailed look at how the models worked especially when the data is not balanced. For example the Naïve Bayes model was better at predicting the metal_can class, than the KNN model.

Performance Comparison:

Accuracy:

The K-Nearest Neighbors model got 31% of the answers on the test.

The Naïve Bayes model also got 31% of the answers on the test.

So both models are the same when it comes to accuracy.. Accuracy is not the only thing that matters. It does not tell us the story, especially when the data is not balanced.

Confusion Matrix:

The confusion matrix for both models shows us how well each one did with each class. The numbers on the diagonal are the ones that were classified correctly. The numbers that are not on the diagonal are the ones that were classified incorrectly.

K-Nearest Neighbors model Confusion Matrix:

The K-Nearest Neighbors model did a job with the metal_can class but it had trouble with the wooden_block and rubber_ball classes.

Naïve Bayes model Confusion Matrix:

The Naïve Bayes model did a job with the metal_can class and it did a more balanced job with all the classes compared to the K-Nearest Neighbors model.

Precision, Recall and F1-Score:

Precision is important because it tells us how many of the things we said were positive are actually positive. This matters when it is expensive to make mistakes.

K-Nearest Neighbors model: The metal_can class had a precision of 0.43, which means it got 43% of the metal_cans right. The plastic_bottle class had a precision of 0.25.

Naïve Bayes model: The metal_can class had a precision of 0.36 and the plastic_bottle class had a precision of 0.35.

Recall is also important because it tells us how many of the things that're actually positive we were able to find. This matters when we do not want to miss anything.

K-Nearest Neighbors model: The metal_can class had a recall of 0.42, which means it found 42% of the metal_cans. The plastic_bottle class had a recall of 0.24.

Naïve Bayes model: The metal_can class had a recall of 0.52, which's better than the K-Nearest Neighbors model.

F1-Score is a way to balance precision and recall especially when the data is not balanced.

K-Nearest Neighbors model: The metal_can class had an F1-score of 0.43 and the plastic_bottle class had an F1-score of 0.24.

Naïve Bayes model: The metal_can class had an F1-score of 0.43 and the plastic_bottle class had an F1-score of 0.32.

Final Comparison:

The K-Nearest Neighbors model did a job with the metal_can class but it did not do as well with the other classes. The Naïve Bayes model did a little overall especially with the metal_can class.

Which Model Performed Better?

The Naïve Bayes model did better with precision and recall for some classes. Even though both models got the accuracy the Naïve Bayes model did better with more classes, especially the metal_can class.

The K-Nearest Neighbors model might be good for some things, like when we have a lot of power to do calculations and the classesre easy to separate.. For this project the Naïve Bayes model was more balanced and reliable.

Robotics Context Integration:

In robotics a robots ability to see and classify objects is crucial for doing tasks on its own. The models made in this project K-Nearest Neighbors (KNN) and Naïve Bayes can be used in a system to help identify objects based on sensor data. This data includes distance, size, color intensity, weight and sensor voltage readings.

The real-world use of this object classification system could be in mobile robots or robotic manipulators used in various industries, such as:

- **Warehouse Automation:** Robots can identify and sort objects, like metal cans and plastic bottles based on sensor data. This enables automated material handling, inventory management and sorting tasks.
- **Recycling Plants:** In a recycling plant robots could use this system to sort types of recyclable materials based on their sensor readings. This could make things more efficient. Reduce human labor.
- **Autonomous Vehicles:** This object classification system could help identify objects on the road like obstacles and pedestrians and make decisions based on their classification.

Constraints and Challenges:

Real-time Processing:

Robotic systems often work in time. This means the classification model needs to make predictions. The models need to be optimized for inference times. If there are delays in prediction it could lead to inefficiencies or even accidents.

KNN can be computationally expensive when dealing with datasets. This is because it requires calculating the distance between the query point and all training data points. For real-time applications optimizing KNN is essential.

Sensor Noise:

Sensor noise is a challenge in real-world robotics. Sensors may provide noisy readings. The models must be robust to noise. One way to handle this is by applying filtering techniques to smooth the sensor data.

Computational Limitations:

Naïve Bayes is a model and well-suited for robots with limited computational resources. However **KNN** may not be suitable for low-resource devices especially if the number of training samples is large.

For systems with computational power like embedded systems Naïve Bayes could be the preferred model due to its simplicity and efficiency.

Scalability:

As the robot interacts with objects the dataset might grow. Models, like KNN could struggle with datasets. To make the system scalable you could explore learning methods. These methods allow the model to learn from data without having to retrain from scratch.

Conclusion:

In conclusion our project successfully created a machine learning solution for classifying objects in systems.

The goal was to design a model that can classify objects based on data from sensors.

To achieve this we. Trained two models: K-Nearest Neighbors (KNN) and Naïve Bayes.

Here are the key findings:

- Both models achieved an accuracy of 31% on the test data.
- Although both models had accuracy Naïve Bayes performed better than KNN in some areas especially for certain objects like metal cans and rubber balls.
- Naïve Bayes was chosen as the model because it worked well across different objects.

Some key points to note:

- K-Nearest Neighbors is a classifier but it can be slow and struggle with large amounts of data.
- Naïve Bayes is fast works with small datasets and is a good choice for applications with limited computing power.

There are areas for future work:

- Improving real-time performance: Both models could be optimized for faster performance in robotic applications.
- Handling data: Adding noise-filtering techniques or using more robust models could improve classification.
- Scalability: As the robot interacts with objects the model must be able to handle the increase efficiently.

Overall our project shows that machine learning models can be used for object classification in robotics.

The models can be. Used in real-world robotic systems.

This can improve automation and decision-making in industries like warehouse automation, recycling and self-driving cars.

The KNN and Naïve Bayes models are promising solutions for object classification, in systems.

These models have the potential to be further developed and integrated into robotic applications.

References:

scikit-learn Documentation:

I used scikit-learn to make the K-Nearest Neighbors and Naïve Bayes models. The scikit-learn documentation is really helpful. It has a lot of examples and explanations for machine learning algorithms.

URL: https://scikit-learn.org/stable/documentation.html

UCI Machine Learning Repository:

The UCI Machine Learning Repository is a place to find datasets for machine learning. It has datasets for lots of things like robotics. Object classification.

URL: https://archive.ics.uci.edu/ml/index.php

Pattern Recognition and Machine Learning by Christopher M. Bishop:

This book is really good for learning about machine learning. It talks about Naïve Bayes and KNN and other algorithms. It also has examples.

ISBN: 978-0387310732

Python Documentation:

I used the Python documentation to learn about Python. It is really helpful for understanding how to use Python.

URL: https://docs.python.org/3/

Machine Learning Yearning by Andrew Ng:

This is a resource for machine learning. It talks about how to make machine learning work well.

URL: https://www.deeplearning.ai/machine-learning-yearning/