Evaluation of a Machine Learning Model

Cat-and-Mouse Game

---

1. Introduction

In this project, we implemented a Q-learning-based model to solve a problem involving a simulated cat-and-mouse game. Subsequently, the model's performance was evaluated and compared to a supervised learning model using Weka, a popular data mining tool.

The primary objective was to analyze the model's ability to predict the mouse's actions (`up`, `down`, `left`, `right`) based on its position and the cat's position in the game environment.

---

2. Procedure

2.1. Data Generation

A Python script was used to generate a dataset based on the mouse's behavior simulation within the game. The dataset included:

- **Attributes:**
  - `mouse_x`, `mouse_y`: Coordinates of the mouse.
  - `cat_x`, `cat_y`: Coordinates of the cat.
  - `action`: Action taken by the mouse (`up`, `down`, `left`, `right`).
- **Instances:** 27,225 examples of mouse decisions under various environment configurations.

The dataset was exported in `.arff` format for compatibility with Weka.

2.2. Pre-analysis of the Dataset

- **Class Imbalance:** The dataset was highly imbalanced:
  - `up`: 25,312 instances (~93%).
  - `down`: 668 instances (~2.5%).
  - `left`: 528 instances (~1.9%).
  - `right`: 717 instances (~2.6%).

This imbalance indicates that the mouse predominantly moves upward more often than in any other direction.

2.3. Model Training in Weka

- **Algorithm Choice:** Since Weka does not have a dedicated algorithm for Q-learning, we selected **J48** (a decision tree implementation of the C4.5 algorithm). J48 was chosen for its interpretability and ability to handle complex decision-making processes.
- **Configuration:**
  - Confidence threshold (`-C`): 0.25.
  - Minimum number of instances per leaf (`-M`): 2.
- **Validation Method:** 10-fold cross-validation was used to evaluate the model's performance.

---

3. Results

3.1. General Metrics

METRIC	VALUE
CORRECTLY CLASSIFIED INSTANCES	94.35%
INCORRECTLY CLASSIFIED INSTANCES	5.65%
KAPPA STATISTIC	0.5101
TIME TO BUILD MODEL	0.74 seconds
NUMBER OF LEAVES	400
SIZE OF THE TREE	799 nodes

3.2. Performance by Class

CLASS	TP RATE	FP RATE	PRECISION	RECALL	F-MEASURE
UP	0.987	0.463	0.966	0.987	0.976
DOWN	0.413	0.008	0.563	0.413	0.477
LEFT	0.273	0.008	0.398	0.273	0.324
RIGHT	0.392	0.008	0.560	0.392	0.461

3.3. Confusion Matrix

ACTUAL \ PREDICTED	UP	DOWN	LEFT	RIGHT
UP	24,986	153	76	97
DOWN	352	276	19	21
LEFT	245	36	144	103
RIGHT	288	25	123	281

---

4. Analysis and Interpretation

4.1. Overall Model Performance

- The model achieved a high overall classification accuracy of **94.35%**, primarily due to the correct classification of the majority class (`up`).
- However, this high accuracy does not reflect balanced performance across all classes, as the model struggled to predict the minority classes (`down`, `left`, `right`).

4.2. Challenges with Minority Classes

- **Classes `down`, `left`, and `right`:**
  - These classes showed low true positive rates (TP Rates): **41.3%** for `down`, **27.3%** for `left`, and **39.2%** for `right`.
  - Precision and recall for these classes were also significantly lower, indicating poor identification of patterns for minority classes.
- **Confusion with the `up` Class:**
  - Many instances of `down`, `left`, and `right` were misclassified as `up`, as shown by the high false positive rate (46.3%) for this class.

4.3. Model Complexity

- The generated decision tree had **400 leaves** and **799 nodes**, indicating a relatively complex model. This complexity could suggest overfitting, especially since the tree is biased towards classifying `up`.

---

5. Conclusion

While the J48-based model performed well overall, its high accuracy was heavily influenced by the overrepresentation of the `up` class. The model's struggles with minority classes highlight the importance of addressing class imbalance for better generalization.