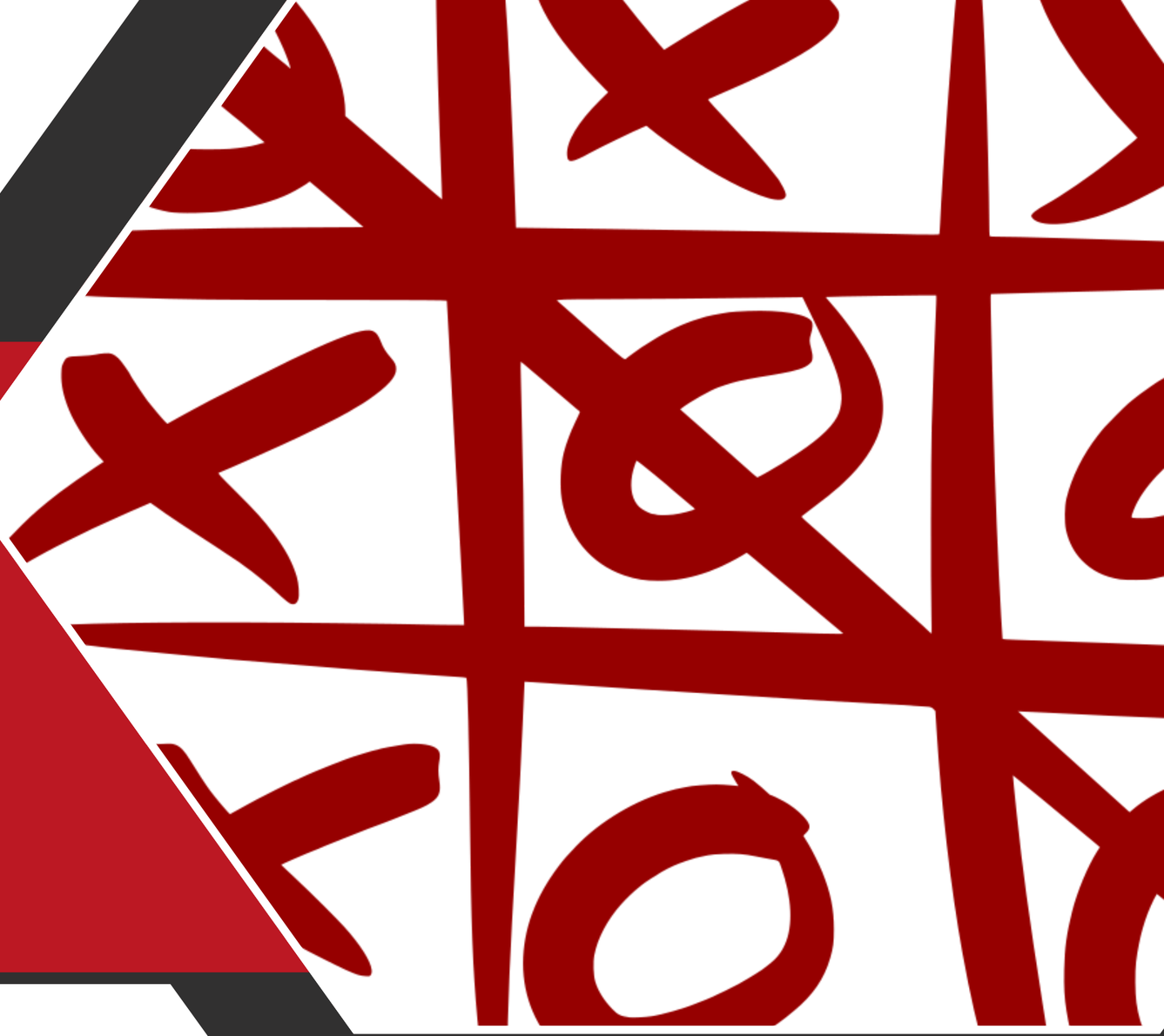
Tic-Tac-Toe Endgame



ที่มาและความสำคัญ

Tic-Tac-Toe เป็นเกมกระดานที่มีกติกาเรียบง่ายและได้รับความนิยมอย่างแพร่หลาย ใช้เป็นเครื่องมือฝึกทักษะเชิงตรรกะและศึกษา ทฤษฎีเกม (Game Theory) รวมถึงปัญญาประดิษฐ์ (AI) รูปแบบการเล่นและกลยุทธ์ของเกมสามารถนำไปประยุกต์ใช้กับอัลกอริทึมของเกมที่ซับซ้อนขึ้น เช่น หมากรุก หรือโกะ อีกทั้งยังเป็นแบบฝึกหัดในการเขียนโปรแกรมเพื่อพัฒนาแนวคิดเชิงตรรกะและการแก้ปัญหา รายงานนี้มุ่งเน้นการศึกษาเขียนโปรแกรม วิเคราะห์กลยุทธ์ และพัฒนา AI สำหรับเกมนี้

วิธีการดำเนินการ

แหล่งที่มา : UCI Machine Learning Repository

01 การวิเคราะห์ข้อมูล

02 โมเดลที่ใช้

03 โค้ดที่ใช้สำหรับสร้างโมเดล

การวิเคราะห์ข้อมูล

01 โครงสร้างข้อมูล

- ข้อมูลมีทั้งหมด 958 แถว และ 10 คอลัมน์
- 9 คอลัมน์แรกเป็นค่าตำแหน่งบนกระดาน ซึ่งมีค่าเป็น
  "x" = ตำแหน่งของผู้เล่น X
  "o" = ตำแหน่งของผู้เล่น O
  "b" = ช่องว่าง (Blank)
  คอลัมน์ที่ 10 เป็นค่าผลลัพธ์ (positive หรือ negative)

02 ตรวจสอบค่าข้อมูลหายและข้อมูลซ้ำ

- ไม่มี missing values ในข้อมูล
- ไม่มี duplicate rows ในข้อมูล

03 วิธีการจัดการข้อมูล

- แปลงค่าหมวดหมู่ (x, o, b) เป็นค่าตัวเลขโดยใช้ Label Encoding
- แยกชุดข้อมูลเป็น Features (X)และ Target (y)
- แบ่งข้อมูลออกเป็น Train (80%) และ Test (20%)

โมเดลที่ใช้

Decision Tree Classifier เนื่องจากโมเดลเหมาะสำหรับการจำแนกข้อมูลประเภท Classification และสามารถจัดการกับข้อมูลที่มีลักษณะ Categorical ได้ดี
เหตุผลที่เลือกใช้โมเดลนี้:

- สามารถทำงานกับข้อมูลที่เป็น หมวดหมู่ (categorical)ได้ดี
- มีความสามารถในการ แยกเงื่อนไขแบบชัดเจน ซึ่งเหมาะกับข้อมูลประเภทเกม
- โครงสร้างของ Decision Treeสามารถแสดงผลลัพธ์เป็นรูปต้นไม้เพื่อการตีความง่าย

โค้ดที่ใช้สำหรับการสร้างโมเดล

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import classification_report,
confusion_matrix

# แปลงข้อมูล categorical เป็นตัวเลข
df_encoded = df.apply(LabelEncoder().fit_transform)

# แยก features และ target
X = df_encoded.iloc[:, :-1]# คอลัมน์ตำแหน่งบนกระดาน
y = df_encoded.iloc[:, -1]# ค่าผลลัพธ์ (positive = 1, negative
= 0)
```

```
# แบ่งข้อมูล train-test
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# สร้างและฝึกโมเดล
model = DecisionTreeClassifier(random_state=42)
model.fit(X_train, y_train)

# ทำนายผล
y_pred = model.predict(X_test)

# แสดงผล Confusion Matrix และ Classification Report
conf_matrix = confusion_matrix(y_test, y_pred)
class_report = classification_report(y_test, y_pred)

print(conf_matrix)
print(class_report)
```

Confusion Matrix

$$\begin{bmatrix} 54 & 13 \\ 2 & 123 \end{bmatrix}$$

01 True Positives (TP) = 123 →โมเดลทำนาย positive ได้ถูกต้อง

02 False Positives (FP) = 13 →โมเดลทำนาย positive ผิดพลาด

03 True Negatives (TN) = 54 →โมเดลทำนาย negative ได้ถูกต้อง

04 ·False Negatives (FN) = 2 →โมเดลทำนาย negative ผิดพลาด

ผลลัพธ์ของโมเดล

Class	Precision	Recall	F1-Score	Support
Nagative (0)	0.96	0.81	0.88	67
Positive (1)	0.90	0.98	0.94	125
Overall accuracy			92%	192

Classification Report

- Accuracy = 92% →โมเดลให้ผลลัพธ์ถูกต้อง 92% ของทั้งหมด
- Precision (ค่าเฉลี่ย) = 93% → ความแม่นยำในการทำนาย positive และ negative
- Recall (ค่าเฉลี่ย) = 90% → โมเดลสามารถจับกลุ่ม positive และ negative ได้ดี
- F1-score (ค่าเฉลี่ย) = 91% → ค่าความสมดุลระหว่าง Precision และ Recall

Class	Precision	Recall	F1-Score	Support
Nagative (0)	0.96	0.81	0.88	67
Positive (1)	0.90	0.98	0.94	125
Overall accuracy			92%	192

การวิเคราะห์ผลลัพธ์ของโมเดล

- Accuracy 92% ถือว่าสูง แต่ต่ำกว่า Random Forest เล็กน้อย
- Recall ของ Positive class เท่ากับ 0.98หมายความว่าโมเดลสามารถตรวจจับ Positive ได้ดี
- Precision ของ Negative class ต่ำกว่าที่คาดไว้ เนื่องจากมี False Positives (13 ครั้ง) และ False Negatives (2 ครั้ง)

Thank you very much!