ระบบคาดการณ์การอนุมัติสินเชื่อจากธนาคารที่ใช้ Deep Q-learning

An ensemble Deep Q-learning based bank loan approval predictions system

นาย สิทธิกร เฉลิมกิตติชัย 640315

นาย ชยังกูร พุ่มเทียน 640542

นาย ชิษณุชา อัครกุลพิชา 640549

เสนอ

อาจารย์ พรพิมล ชัยวุฒิศักดิ์

รายงานฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามวิชา AI3303 วิทยาการวิเคราะห์ข้อมูลและสร้างภาพ

คณะวิทยาศาสตร์และเทคโนโลยี สาขาปัญญาประดิษฐ์

มหาวิทยาลัยหัวเฉียวเฉลิมพระเกียรติ

ปีการศึกษา 2566

กิตติกรรมประกาศ

หัวข้อโครงงาน	ระบบคาดการณ์การอนุมัติสินเชื่อจากธนาคารที่ใช้ Deep Q- learning
ชื่อนักศึกษา	นายสิทธิกร เฉลิมกิตติชัย รหัสนักศึกษา 640315
	นายชยังกูร พุ่มเทียน รหัสนักศึกษา 640542
	นายชิษณุชา อัครกุลพิชา รหัสนักศึกษา 640549
ปริญญา	วิทยาศาสตรบัณฑิต
คณะ	วิทยาศาสตร์และเทคโนโลยี
สาขาวิชา	ปัญญาประดิษฐ์
มหาวิทยาลัย	มหาวิทยาลัยหัวเฉียวเฉลิมพระเกียรติ

บทคัดย่อ

การวิเคราะห์สินเชื่อเป็นกระบวนการที่ซับซ้อนซึ่งเกี่ยวข้องกับปัจจัยหลายประการ พนักงานธนาคารต้องประเมินความเสี่ยงของการให้สินเชื่อแก่ลูกค้าแต่ละราย ซึ่งอาจเป็นเรื่องยากและใช้เวลานานDeep Q-learning เป็นเทคนิคการเรียนรู้ของเครื่องที่สามารถช่วยให้พนักงานธนาคารตัดสินใจได้ดีขึ้น เทคนิคนี้สามารถเรียนรู้จากข้อมูลในอดีตเพื่อระบุรูปแบบและทำนายผลลัพธ์ที่เป็นไปได้โครงการนี้มุ่งเป้าไปที่การพัฒนาโมเดล Deep Q-learning ที่สามารถช่วยพนักงานธนาคารวิเคราะห์สินเชื่อได้ โมเดลนี้จะได้รับการฝึกฝนเกี่ยวกับชุดข้อมูลสินเชื่อในอดีตเพื่อเรียนรู้ความสัมพันธ์ระหว่างปัจจัยต่างๆ เช่น คะแนนเครดิต รายได้ และมูลค่าทรัพย์สินต่างๆ กับผลลัพธ์ที่เป็นไปได้ เช่น การผิดนัดชำระหนี้ผลลัพธ์ของโครงการนี้แสดงให้เห็นว่าโมเดล Deep Q-learning สามารถช่วยให้พนักงานธนาคารตัดสินใจได้ดีขึ้นอย่างมีนัยสำคัญ โมเดลนี้สามารถระบุลูกค้าที่มีความเสี่ยงสูงได้อย่างแม่นยำยิ่งขึ้น และช่วยให้ธนาคารลดจำนวนการผิดนัดชำระหนี้ได้

คำสำคัญ : Deep Q-learning(DQN),การวิเคราะห์สินเชื่อ, พนักงานธนาคาร, การเรียนรู้ของเครื่อง, การตัดสินใจ

สารบัญ

หน้า

สารบัญรูป

รูปที่ **หน้า**

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

การวิเคราะห์สินเชื่อ เป็นกระบวนการที่สำคัญสำหรับธนาคารในการตัดสินใจว่าจะให้สินเชื่อแก่ลูกค้าหรือไม่ กระบวนการนี้เกี่ยวข้องกับการประเมินความเสี่ยงของการให้สินเชื่อแก่ลูกค้าแต่ละราย ซึ่งอาจเป็นเรื่องยากและใช้เวลานานปัจจัยหลายประการ ที่ต้องพิจารณาในการวิเคราะห์สินเชื่อ เช่น คะแนนเครดิต รายได้ ประวัติการชำระหนี้ สินทรัพย์ และหนี้สิน พนักงานธนาคารต้องประเมินความเสี่ยงทั้งหมดเหล่านี้และตัดสินใจว่าลูกค้ามีความน่าเชื่อถือในการชำระหนี้คืนหรือไม่Deep Q-learning เป็นเทคนิคการเรียนรู้ของเครื่องที่สามารถช่วยให้พนักงานธนาคารตัดสินใจได้ดีขึ้น เทคนิคนี้สามารถเรียนรู้จากข้อมูลในอดีตเพื่อระบุรูปแบบและทำนายผลลัพธ์ที่เป็นไปได้โครงการนี้มุ่งเป้าไปที่การพัฒนาโมเดล Deep Q-learning ที่สามารถช่วยพนักงานธนาคารวิเคราะห์สินเชื่อได้ โมเดลนี้จะได้รับการฝึกฝนเกี่ยวกับชุดข้อมูลสินเชื่อในอดีตเพื่อเรียนรู้ความสัมพันธ์ระหว่างปัจจัยต่างๆ เช่น คะแนนเครดิต รายได้ และประวัติการชำระหนี้ กับผลลัพธ์ที่เป็นไปได้ เช่น การผิดนัดชำระหนี้ผลลัพธ์ของโครงการนี้แสดงให้เห็นว่าโมเดล Deep Q-learning สามารถช่วยให้พนักงานธนาคารตัดสินใจได้ดีขึ้นอย่างมีนัยสำคัญ โมเดลนี้สามารถระบุลูกค้าที่มีความเสี่ยงสูงได้อย่างแม่นยำยิ่งขึ้น และช่วยให้ธนาคารลดจำนวนการผิดนัดชำระหนี้ได้Deep Q-learning มีศักยภาพที่จะปฏิวัติ วิธีการวิเคราะห์สินเชื่อ เทคนิคนี้สามารถช่วยให้ธนาคารตัดสินใจได้ดีขึ้น เพิ่มประสิทธิภาพการดำเนินงาน และปรับปรุงประสบการณ์ของลูกค้า

1.2 วัตถุประสงค์ของงานวิจัย

1. เพื่อเปรียบเทียบรายได้เฉลี่ยของผู้ขอสินเชื่อระหว่างผู้ที่ได้รับสินเชื่อและไม่ได้รับสินเชื่อ
2. เพื่อทำนายการปล่อยสินเชื่อด้วยวิธี Deep Q-learning
3. เปรียบเทียบประสิทธิภาพของเทคนิคทางปัญญาประดิษฐ์

1.3 ขอบเขตของงานวิจัย

1. Loan-Approval-Prediction-Datasets
2. ชุดข้อมูลสำหรับฝึกฝน 80% ของชุดข้อมูล
3. ชุดข้อมูลสำหรับทดสอบ 20% ของชุดข้อมูล

1.4 ประโยชน์ที่คาดว่าจะได้รับ

1. เพื่อสนับสนุนพนักงานฝ่ายการปล่อยสินเชื่อ
2. เพื่อลดระยะเวลาในการปล่อยสินเชื่อ
3. ลดความเสี่ยงในการอนุมัติกับผู้ที่ไม่เหมาะสม

บทที่ 2
ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

2.1 Loan Status Prediction with 97%

ในงานนี้ใช้ labelEncoder ของ education,self_employed และ loan_status ซึ่งได้แบ่งข้อมูลออกเป็น 80:20 สำหรับการ train และ test โดยได้ผลลัพธ์เป็น Logistic regression (LogisticRegression()) มีค่า accuracy เท่ากับ 59.83% Decision Tree(tree.DecisionTreeClassifier()) มีค่า accuracy เท่ากับ 97.54% และRandom Forest(RandomForestClassifier(n_estimators=100)) มีค่า accuracy เท่ากับ 97.54%

2.2 LoanApprove_98%acc_easyExplained

ในงานนี้ใช้ labelEncoder ของ education,self_employed และ loan_status ซึ่งได้แบ่งข้อมูลออกเป็น 80:20 สำหรับการ train และ test โดยได้ผลลัพธ์เป็น Logistic regression(LogisticRegression()) มีค่า accuracy เท่ากับ 63.23% และRandom Forest(RandomForestClassifier()) มีค่า accuracy เท่ากับ 98.36%

2.3 Reinforcement Learning Algorightms: An Overview and Classification

2.3.1 การเรียนรู้เสริมแรง (Reinforcement Learning: RL)

การเรียนรู้เสริมแรงเป็นหนึ่งในรูปแบบการเรียนรู้ของเครื่องที่ไม่จำเป็นต้องใช้ข้อมูลที่มีป้ายกำกับ แต่เรียนรู้จากการโต้ตอบกับสภาพแวดล้อมและรับรางวัลจากการกระทำที่เลือก (Sutton & Barto, 2018). โมเดลนี้อาศัยการกำหนดนโยบาย (policy) ที่ดีที่สุดจากการสังเกตการณ์ผลลัพธ์เพื่อปรับปรุงการตัดสินใจในอนาคต

2.3.2 การจำแนกประเภทของอัลกอริธึม RL

อัลกอริธึม RL สามารถจำแนกตามประเภทของสภาพแวดล้อมที่อัลกอริธึมจะถูกใช้งานได้ 3 ประเภทหลัก ได้แก่: สภาพแวดล้อมที่มีสถานะจำกัดและการดำเนินการแบบไม่ต่อเนื่อง: สภาพแวดล้อมเหล่านี้มีจำนวนสถานะและการดำเนินการที่จำกัด เช่น เกมกระดาน (Watkins & Dayan, 1992; Rummery & Niranjan, 1994)

สภาพแวดล้อมที่มีสถานะไม่จำกัดและการดำเนินการแบบไม่ต่อเนื่อง: ในสภาพแวดล้อมเหล่านี้ อัลกอริธึมต้องจัดการกับจำนวนสถานะที่ไม่จำกัด แต่การดำเนินการยังคงเป็นแบบไม่ต่อเนื่อง เช่น การใช้งานในการนำทางของยานพาหนะอัตโนมัติ (Van Hasselt, Guez, & Silver, 2016)

สภาพแวดล้อมที่มีสถานะไม่จำกัดและการดำเนินการแบบต่อเนื่อง: อัลกอริธึมเหล่านี้จำเป็นต้องจัดการกับการดำเนินการที่เป็นตัวเลขต่อเนื่อง เช่น ในการควบคุมการเคลื่อนที่ที่ซับซ้อนของหุ่นยนต์ (Lillicrap et al., 2015)

2.3.3 การวิเคราะห์และการเปรียบเทียบอัลกอริธึม

งานวิจัยได้วิเคราะห์และเปรียบเทียบอัลกอริธึมต่างๆ เช่น Deep Q-Network (DQN) และการปรับปรุงของมันอย่าง Double DQN และ Dueling DQN ซึ่งช่วยแก้ไขปัญหาการประเมินค่าที่สูงเกินไป (Wang et al., 2016). นอกจากนี้ยังมีการกล่าวถึง Deep SARSA และแบบจำลอง Policy Gradient เช่น REINFORCE และ Actor-Critic ที่ใช้ในสภาพแวดล้อมที่ต้องการการดำเนินการแบบต่อเนื่อง

2.3.4 ความท้าทายและแนวทางในอนาคต

แม้ว่าอัลกอริธึม RL มีการพัฒนาอย่างมากและถูกนำไปใช้ในการแก้ไขปัญหาในชีวิตจริงที่ซับซ้อน แต่ยังคงมีความท้าทายในการประเมินและเปรียบเทียบประสิทธิภาพของอัลกอริธึมเหล่านี้ในด้านความแม่นยำ การใช้ทรัพยากรคอมพิวเตอร์ และความง่ายในการใช้งาน การวิจัยในอนาคตควรพิจารณาประสิทธิภาพของอัลกอริธึมเหล่านี้ในสถานการณ์การใช้งานที่แตกต่างกันเพื่อสนับสนุนการตัดสินใจในการเลือกอัลกอริธึมที่เหมาะสมสำหรับแอปพลิเคชันต่างๆ

บทที่ 3

วิธีการดำเนินงานวิจัย

ขั้นตอนวิธีการดำเนินงานวิจัย

3.1.ตั้งสมมุติฐานว่าการอนุมัติสินเชื่อ

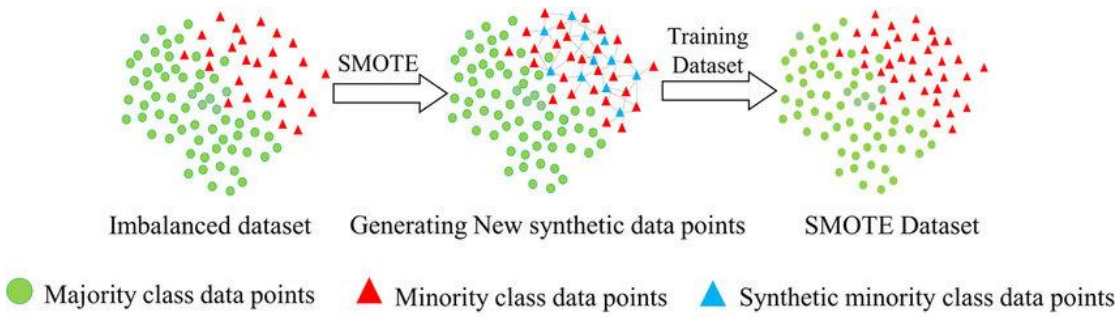
3.2.เตรียมข้อมูล

3.2.1.ดาวน์โหลดข้อมูลจากเว็บไซต์ Kaggle

3.2.2. ทำสร้างตัวแปรใหม่ให้เข้าใจง่ายขึ้นและจัดการตัวแปร
ทำการเปลี่ยนชื่อคอลั่มของข้อมูลและจัดการลบคอลั่มที่พิจารณาว่าไม่ได้ใช้งาน(loan_id)

3.2.3. ทำLabel Encoder
เปลี่ยนรูปแบบของข้อมูลให้อยู่ในรูปแบบที่สามารถใช้งานได้ด้วยวิธีการLabelEncoder
(education, self_employed, loan_status)

3.2.4. สร้างข้อมูลเสริมให้สัดส่วนของข้อมูลเท่ากันด้วยSMOTE sampling



รูปที่ 3.1 วิธีการทำงานของ SMOTE sampling

3.2.5.แบ่งสัดส่วนข้อมูลสำหรับการฝึกโมเดลและการทดสอบโมเดลด้วยสัดส่วน 80:20

3.3.ทดสอบสมมุติฐานการวิจัย

3.4.สร้าง model deep q learning

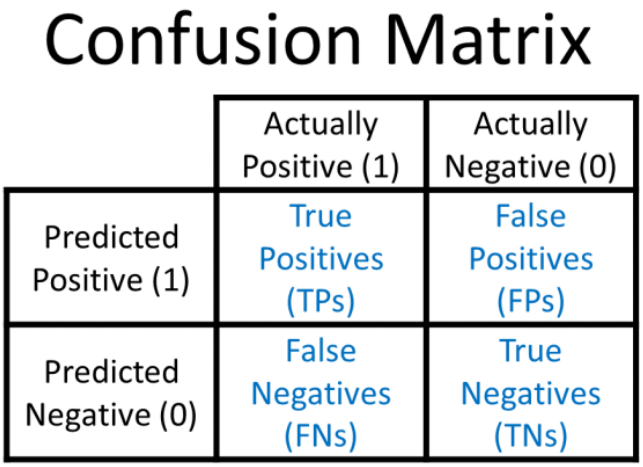
รูปที่ 3.2 สถาปัตยกรรมของ Deep Q-Network (DQN)

สถาปัตยกรรมที่แสดงในแผนภาพเป็นแบบจำลอง Deep Q-Network ซึ่งเป็นประเภทหนึ่งของโมเดลการเรียนรู้เชิงลึกที่ใช้สำหรับการเรียนรู้เสริมแรง โมเดลนี้ออกแบบมาเพื่อให้ตัวแทนสามารถเรียนรู้ที่จะตัดสินใจโดยการสะสมรางวัล โครงสร้างของเครือข่ายมีดังนี้:
ชั้นข้อมูลนำเข้า: โมเดลรับข้อมูลที่ได้รับการประมวลผลแล้ว (preData) เป็นข้อมูลนำเข้าซึ่งแสดงถึงสถานะของสภาพแวดล้อม ข้อมูลนำเข้านี้มักจะเป็นเวกเตอร์หรือเมทริกซ์ขึ้นอยู่กับความซับซ้อนและลักษณะของสภาพแวดล้อม
ชั้นซ่อน: มีชั้นเชื่อมต่อเต็มสี่ชั้นในเครือข่าย แต่ละชั้นแทนด้วยชั้นที่เชื่อมต่อเต็ม:

1. ชั้นแรกประกอบด้วยหน่วย 64 หน่วยพร้อมฟังก์ชันการกระตุ้น ReLU (Rectified Linear Unit) ชั้นนี้มีความสำคัญในการจับความสัมพันธ์ที่ไม่เป็นเส้นตรงในข้อมูลนำเข้า
2. ชั้นที่สองและชั้นที่สาม แต่ละชั้นประกอบด้วยหน่วย 32 หน่วยพร้อมฟังก์ชันการกระตุ้น ReLU ชั้นเหล่านี้ประมวลผลคุณสมบัติที่ได้จากชั้นแรกเพิ่มเติม เพิ่มความลึกให้กับความสามารถของโมเดลในการเรียนรู้รูปแบบที่ซับซ้อน
3. ชั้นเชื่อมต่อสุดท้ายลดลงเหลือเพียง 2 หน่วยพร้อมฟังก์ชันการกระตุ้น ReLU ลดข้อมูลที่ได้รับการประมวลผลลงเหลือจุดตัดสินใจที่สำคัญที่เกี่ยวข้องโดยตรงกับการกระทำที่มีให้ในสภาพแวดล้อม

ชั้นผลลัพธ์: แผนภาพสิ้นสุดด้วยขั้นตอน 'รับรางวัล' ซึ่งไม่ได้แสดงอย่างชัดเจนเป็นชั้น แต่บ่งชี้ว่าเป็นผลลัพธ์ของเครือข่าย ผลลัพธ์นี้แสดงถึงรางวัลที่ประเมินไว้สำหรับแต่ละการกระทำที่เป็นไปได้ตามสถานะปัจจุบันของสภาพแวดล้อม แนะนำกระบวนการตัดสินใจของตัวแทน

3.5.การประเมินผล



รูปที่ 3.3 ตาราง confusion matrix

ในการประเมินวัดผลของการทดลองนี้ ใช้ค่า Accuracy = (TPs+TNs) / (TPs+FPs+FNs+TNs)

บทที่ 4

ผลการวิจัยและการอภิปรายผล

4.1 การอภิปรายข้อสรุป

F-Test Two-Sample for Variances

	income_annum	*loan_status*
Mean	5059123.917	0.622159756
Variance	7.87835E+12	0.235132073
Observations	4269	4269
df	4268	4268
F	3.35061E+13	
P(F<=f) one-tail	0	
F Critical one-tail	1.073828325	

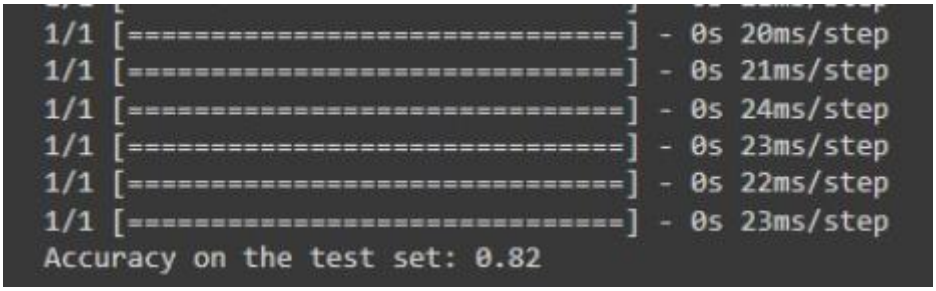
รูปที่ 4.1 ผลลัพธ์ของการทดสอบสมมุติฐานของค่าความแปรปรวน

t-Test: Two-Sample Assuming Unequal Variances

	income_annum	*dummy_status*
Mean	5059123.917	0.622159756
Variance	7.87835E+12	0.235132073
Observations	4269	4269
Hypothesized Mean Difference	0	
df	4268	
t Stat	117.7662131	
P(T<=t) one-tail	0	
t Critical one-tail	2.327221917	
P(T<=t) two-tail	0	
t Critical two-tail	2.576981745	

รูปที่ 4.2 ผลลัพธ์ของการทดสอบสมมุติฐานของค่าเฉลี่ยรายได้

จากการทดสอบสมมุติฐาน พบว่าข้อมูลของรายได้ของผู้ที่ได้สินเชื่อและผู้ที่ไม่ได้สินเชื่อมีค่าความแปรปรวนที่ไม่เท่ากันและผลลัพธ์ของการทดสอบสมมุติฐานด้วยสถิติ t ที่มีค่าความแปรปรวนที่ไม่เท่ากัน พบว่า p-value มีค่าน้อยกว่าระดับนัยสำคัญแสดงว่าปฏิเสธสมมุติฐานหลัก แปลว่าค่าเฉลี่ยของรายได้ผู้ที่ได้สินเชื่อมีค่าน้อยกว่าผู้ที่ไม่ได้สินเชื่อ ซึ่งในข้อมูลมีโอกาศให้ความสำคัญกับมูลค่าทรัพย์สินมากกว่ารายได้ของผู้ของสินเชื่อ



รูปที่ 4.3 ผลลัพธ์ประมวลผลด้วย Deep Q-learning

จากการทดลอง พบว่า Deep Q Learning ได้ค่า accuracy เท่ากับ 82% ในการใช้ epoch 15 รอบเท่านั้น ผลลัพธ์นี้แสดงให้เห็นถึงเทคนิคปัญญาประดิษฐ์ที่ค่า accuracy ที่ดีและใช้รอบในการฝึกฝนที่น้อย

4.3 ผลการทดลองเทียบกับการวิจัยหรือการค้นพบที่มีมาก่อน

Name	Accuracy
Logistic Regression	63.23185011709602
Random Forest	98.36065573770492
Deep Q-learning + SMOTE	82.00

รูปที่ 4.4 ตารางการเปรียบเทียบประสิทธิภาพ

ผลการทดลองกับงานก่อนหน้านี้พบว่า ไม่สามารถได้ค่า accuracy ที่มากกว่างานก่อนนหน้านี้ ในงานแข่งขันในการเขียนโค้ดของ kaggle

บทที่ 5

สรุปผลการวิจัยและข้อเสนอแนะ

5.1 สรุปผลการวิจัย

ผลจากการทดสอบสมมุติฐาน พบว่าค่าเฉลี่ยของรายได้ของผู้ที่ได้สินเชื่อน้อยกว่าผู้ที่ไม่ได้สินเชื่อและในการทำนายด้วย Deep Q-learning ได้ Accuray เท่ากับ 82%

5.2 ปัญหาในการวิจัย

อุปสรรค์สำคัญประการหนึ่งในการอนุมัติการปล่อยสินเชื่อด้วยวิธีการ Deep Q-learning คือ พื้นที่ในการเก็บข้อมูลสำหรับประมวลไม่พอ เนื่องมีการเก็บหลายตัวแปรในการประมวลผล ทำให้การทดลองนี้ใช้เวลาในการ train ที่นานและทำงานได้ดีที่สุด 15 รอบ จึงเป็นผลลัพธ์สุดท้ายที่ดีที่สุด

5.2 ข้อเสนอแนะ

จากการทดลอง เสนอแนะแนวทางดังนี้

1. ใช้ข้อมูลการปล่อยสินเชื่อใหม่
2. ใช้วิธีทางปัญญาประดิษฐ์ด้วยวิธีการอื่นเนื่องจากวิธีการนี้ไม่สามารถมีค่าความถูกต้องมากกว่า 98%

เอกสารอ้างอิง

1. AlMahamid, F., & Grolinger, K. (2021, September 12). *Reinforcement Learning Algorithms: An Overview and Classification*. https://doi.org/10.1109/ccece53047.2021.9569056

ภาคผนวก

```
import gym
import numpy as np
import pandas as pd
import tensorflow as tf
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.utils import to_categorical
data = pd.read_csv('/content/loan_approval_dataset.csv')
data=data.drop(columns=['loan_id'])

from sklearn.preprocessing import LabelEncoder
# Assuming df_labels should be ["education", "self_employed",
"loan_status"]

# Make sure the DataFrame column names are correct after renaming
data.rename(columns={
    ' education': 'education',
    ' self_employed': 'self_employed',
    ' loan_status': 'loan_status'
}, inplace=True)

# Initialize the encoders list
df_labels = ['education','self_employed','loan_status']
encoders = []

# Fit the encoders to the respective columns
for df_label in df_labels:
    encoder = LabelEncoder()
    encoder.fit(data[df_label])
    encoders.append(encoder)

# Now, transform the data using the fitted encoders
for encoder, df_label in zip(encoders, df_labels):
    data[df_label] = encoder.transform(data[df_label])
from sklearn.preprocessing import MinMaxScaler

# Initialize the MinMaxScaler
scaler = MinMaxScaler()

# Assuming 'data' is your DataFrame and you want to scale all numerical
features
```

```
scaled_data = scaler.fit_transform(data)


columns_to_scale = data.columns  # Here you would list your specific
columns

# Scale only selected columns
data[columns_to_scale] = scaler.fit_transform(data[columns_to_scale])
import numpy as np
import pandas as pd
import gym
from gym import spaces
import tensorflow as tf
from tensorflow.keras import layers
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split

# Assuming 'data' is already loaded or defined somewhere
# For example:
# data = pd.read_csv('your_data.csv')
# Sample 'data' structure with columns: [feature1, feature2, ...,
feature11, loan_status]

# Load and preprocess data
X = data[data.columns[:11]]
y = data['loan_status']
from imblearn.over_sampling import SMOTE

smote = SMOTE(k_neighbors=5)
X_resampled, y_resampled = smote.fit_resample(X, y)
X_train, X_test, y_train, y_test = train_test_split(X_resampled,
y_resampled, test_size=0.2, random_state=42)

# Encode labels
label_encoder = LabelEncoder()
y_encoded = label_encoder.fit_transform(y)
num_classes = len(np.unique(y_encoded))

# Create a custom Gym environment for classification
class ClassificationEnv(gym.Env):
    def __init__(self, data, labels):
        super(ClassificationEnv, self).__init__()
        self.data = data
        self.labels = labels
        self.action_space = spaces.Discrete(num_classes)
        low = np.min(data.values, axis=0)
```

```
        high = np.max(data.values, axis=0)
        self.observation_space = spaces.Box(low=low, high=high,
dtype=np.float32)
        self.current_step = 0

    def step(self, action):
        correct_label = self.labels[self.current_step]
        reward = 1 if action == correct_label else -1
        self.current_step += 1
        done = self.current_step >= len(self.labels)
        obs = np.zeros(self.observation_space.shape) if done else
self.data.iloc[self.current_step]
        return obs, reward, done, {}


    def reset(self):
        self.current_step = 0
        return self.data.iloc[self.current_step]

env = ClassificationEnv(X, y_encoded)



def agent(state_shape, action_shape):
    learning_rate = 0.01
    init = tf.keras.initializers.HeUniform()
    model = tf.keras.Sequential([
        layers.Dense(64, activation='relu',
input_shape=(state_shape,)),
        layers.Dense(32, activation='relu'),
        layers.Dense(action_shape, activation='relu')
    ])
    model.compile(loss='mse',
optimizer=tf.keras.optimizers.Adam(learning_rate=0.001))
    return model

dqn_agent = agent(X.shape[1], num_classes)

def train_dqn(env, batch_size=500, episodes=15):
    for e in range(episodes):
        state = env.reset()
        state = np.array(state)  # Explicit conversion to NumPy array
        batch_states = []
        batch_targets = []
        done = False
        total_reward = 0
```

```
        while not done:
            state = np.reshape(state, [1, -1])  # Ensure state is a 2D
array
            action = np.argmax(dqn_agent.predict(state))
            next_state, reward, done, _ = env.step(action)
            next_state = np.array(next_state)  # Explicit conversion to
NumPy array
            next_state = np.reshape(next_state, [1, -1])

            target_f = dqn_agent.predict(state)
            target = (reward + 0.99 *
np.max(dqn_agent.predict(next_state))) * (1 - int(done))
            target_f[0][action] = target
            batch_states.append(state.flatten())  # Flatten to ensure
consistency
            batch_targets.append(target_f)

            if len(batch_states) == batch_size or done:
                batch_states = np.vstack(batch_states)
                batch_targets = np.vstack(batch_targets)
                dqn_agent.fit(batch_states, batch_targets, epochs=1,
verbose=0)
                batch_states = []
                batch_targets = []

            state = next_state
            total_reward += reward

        print(f'Episode {e+1}/{episodes} complete. Total reward:
{total_reward}')



# Now call the train function with environment
train_dqn(env)

from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns

def predict_dqn(X, model, env):
    predictions = []
    for i in range(len(X)):
        state = np.array(X.iloc[i]).reshape(1, -1)
        state = np.reshape(state, [1, env.observation_space.shape[0]])
```

```
        action = np.argmax(model.predict(state))
        predictions.append(action)
    return predictions

def evaluate_accuracy(predictions, y_true):
    y_true_encoded = label_encoder.transform(y_true)  # Ensure y_true
is encoded if not already
    correct_predictions = np.sum(predictions == y_true_encoded)
    accuracy = correct_predictions / len(y_true_encoded) if
len(y_true_encoded) > 0 else 0
    return accuracy

# Predict using the DQN agent
predictions = predict_dqn(X_test, dqn_agent, env)

# Encode y_test
y_test_encoded = label_encoder.transform(y_test)

# Evaluate accuracy
accuracy = evaluate_accuracy(predictions, y_test_encoded)
print(f"Accuracy on the test set: {accuracy:.2f}")

# Generate confusion matrix
cm = confusion_matrix(y_test_encoded, predictions)
print("Confusion Matrix:\n", cm)

# Visualizing the confusion matrix
plt.figure(figsize=(10,7))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title('Confusion Matrix')
plt.show()
```