# การทำนายการซื้อซ้ำของผู้ซื้อโดยใช้เทคนิคการเรียนรู้ของเครื่องจักร

# REPEAT BUYER PREDICTION USING MACHINE LEARNING TECHNIQUE

ธนัท จรณะสมบูรณ์

บัณฑิตวิทยาลัยมหาวิทยาลัยศรีนครินทรวิโรฒ

2561

# การทำนายการซื้อซ้ำของผู้ซื้อโดยใช้เทคนิคการเรียนรู้ของเครื่องจักร

ธนัท จรณะสมบูรณ์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร
วิทยาศาสตรมหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ
คณะวิทยาศาสตร์ มหาวิทยาลัยศรีนครินทรวิโรฒ
ปีการศึกษา 2561

# REPEAT BUYER PREDICTION USING MACHINE LEARNING TECHNIQUE

THANAT CHARANASOMBOON

A Thesis Submitted in partial Fulfillment of Requirements

for MASTER OF SCIENCE (Information Technology)

Faculty of Science Srinakharinwirot University

2018

ปริญญานิพนธ์

เรื่อง

การทำนายการซื้อซ้ำของผู้ซื้อโดยใช้เทคนิคการเรียนรู้ของเครื่องจักร

ของ

ธนัท จรณะสมบูรณ์

ได้รับอนุมัติจากบัณฑิตวิทยาลัยให้นับเป็นส่วนหนึ่งของการศึกษาตามหลักสูตร

ปริญญาวิทยาศาสตรมหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ

ของมหาวิทยาลัยศรีนครินทรวิโรฒ

..... คณบดีบัณฑิตวิทยาลัย

(รองศาสตราจารย์ นายแพทย์ฉัตรชัย เอกปัญญาสกุล)

.....

คณะกรรมการสอบปากเปล่าปริญญานิพนธ์

..... ที่ปรึกษาหลัก

(ผู้ช่วยศาสตราจารย์ ดร.วราภรณ์ วิยานนท์)

..... ประธาน

(ผู้ช่วยศาสตราจารย์ ดร.อัครา ประโยชน์)

..... กรรมการ

(อาจารย์ ดร.ศิริสรรพ เหล่าหะเกียรติ)

| | |
|---|---|
| ชื่อเรื่อง | การทำนายการซื้อซ้ำของผู้ซื้อโดยใช้เทคนิคการเรียนรู้ของเครื่องจักร |
| ผู้วิจัย | ธนัท จรณะสมบูรณ์ |
| ปริญญา | วิทยาศาสตรมหาบัณฑิต |
| ปีการศึกษา | 2561 |
| อาจารย์ที่ปรึกษา | ผู้ช่วยศาสตราจารย์ ดร. วราภรณ์ วิยานนท์ |

เนื่องจากธุรกิจร้านค้านำเสนอโปรโมชั่นส่วนลดต่าง ๆ เพื่อดึงดูดลูกค้าที่มีความภักดีต่อธุรกิจร้านค้าและกลับมาซื้อสินค้าในอนาคต ลูกค้าที่ใช้โปรโมชั่นสามารถจำแนกได้เป็นสองประเภทคือ (1) ลูกค้าที่ใช้โปรโมชั่นเพื่อซื้อสินค้าเพียงครั้งเดียว (one-time deal hunter) (2) ลูกค้าที่กลับมาซื้อสินค้าซ้ำ (loyal customer หรือ repeat buyer) โดยลูกค้าที่ใช้โปรโมชั่นเพื่อซื้อสินค้าเพียงครั้งเดียวทำให้โปรโมชั่นที่จัดขึ้นไม่มีเกิดความคุ้มค่า การสร้างแบบจำลองการทำนายการซื้อซ้ำของผู้ซื้อจึงเป็นเครื่องมือที่ได้รับความนิยมเพื่อที่จะใช้ในการทำนายว่าลูกค้าคนใดที่จะเป็นลูกค้าที่มีความภักดีต่อธุรกิจร้านค้าหลังจากที่ได้รับโปรโมชั่นไปแล้วเพื่อให้ได้กลุ่มลูกค้าที่แม่นยำและเพิ่มความคุ้มค่าในการจัดทำโปรโมชั่น

ในงานวิจัยนี้ผู้วิจัยได้เสนอวิธีการสร้างแบบจำลองการทำนายการกลับมาซื้อซ้ำของผู้ซื้อเพื่อจำแนกลูกค้าที่จะกลับมาซื้อซ้ำในอนาคตสำหรับสร้างรายได้ให้กับร้านค้า และช่วยให้ธุรกิจร้านค้าลดต้นทุนในการจัดทำโปรโมชั่นในอนาคต โดยในงานวิจัยนี้ผู้วิจัยใช้ชุดข้อมูลจาก Kaggle “Acquire Valued Shopper Challenge” โดยนำข้อมูลประวัติการซื้อสินค้าและข้อมูลโปรโมชั่นที่ลูกค้าได้รับ ด้วยการสร้างฟีเจอร์จากการจัดกลุ่มผลรวมข้อมูลประวัติการซื้อสินค้าและโปรโมชั่นในช่วงระยะเวลาต่าง ๆ โดยใช้เทคนิคการจำแนกข้อมูล (Classification) และการวิเคราะห์การถดถอย (Regression) ในการสร้างแบบจำลองการทำนายได้แก่ Random forest classifier, Random forest regressor, XGBoost และ Gradient Boost ร่วมกับเทคนิค Leave One Out และวัดผลประสิทธิภาพการทำนายด้วยพื้นที่ใต้โค้ง ROC (Area Under the operating characteristic Curve หรือ AUC) โดยได้ผลลัพธ์สุดท้ายคือ 0.60936 หรือเทียบเท่ากับอันดับที่ 20 ของการแข่งขัน โดยคะแนนของผู้ชนะคือ 0.626703

คำสำคัญ : การทำนายการซื้อซ้ำ, เทคนิคการเรียนรู้ของเครื่อง

| | |
|---|---|
| Title | REPEAT BUYER PREDICTION USING MACHINE LEARNING TECHNIQUE |
| Author | THANAT CHARANASOMBOON |
| Degree | MASTER OF SCIENCE |
| Academic Year | 2018 |
| Thesis Advisor | Assistant Professor Dr. Waraporn Viyanon |

Many consumer brands try their best to offer promotions that attract new customers and the hope that customers will remain loyal to the brand and come back to buy more. Customers who use promotions can be classified into two categories: (1) customers bought products because they had a promotion (one-time deal hunter); (2) customers who come back to buy more products after the promotion period (loyal customers or repeat buyers); customers who use the promotions to buy products only once make the promotion seem less worthwhile. Thus, the prediction model is now a popular tool to make predictions about customers who remained loyal after the promotional period and to make it more targeted to maximize its effectiveness. In this paper, a solution is proposed for repeat buyer predictions in order to identify buyers with the potential to come back to buy more products. This solution would help companies reduce their budgets in terms of distributing promotional offers to customers, not just one-time deal hunters. In this study, the dataset used was called "Acquire Value Shopper Challenge", which focused on customers transactions and used promotion offering information on the Kaggle website. The aspect of feature engineering was based on a summary and the aggregation of customers transactions over a few months. The classifier and regressor techniques for creating prediction models, which included a random forest classifier, a random forest regressor, an XGBoost classifier, and a gradient boost regressor, which were incorporated with Leave one out technique. To evaluate the models, the area under the operating characteristic curve (AUC) was used. The final result was 0.60936 which was equivalent to twentieth place in the contest. The first place in the contest was 0.62703.

Keyword : Machine learning techniques, Repeat Buyer Predictions

# กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จลุล่วงได้ ด้วยความกรุณาจากคณาจารย์ทุกท่านที่ให้คำปรึกษา ผู้ช่วยศาสตราจารย์ ดร.วราภรณ์ วิยานนท์และผู้ช่วยศาสตราจารย์ ดร.ประดิษฐ์ มิตราปิยานุรักษ์ ที่กรุณาเป็นที่ปรึกษาและให้ความช่วยเหลือ คำแนะนำ ทำให้ปริญญานิพนธ์ฉบับนี้เสร็จสมบูรณ์

ขอขอบพระคุณทุนสนับสนุนการเข้าร่วมประชุมและนำเสนอผลงานทางวิชาการจากบัณฑิตวิทยาลัย มหาวิทยาลัยศรีนครินทรวิโรฒ ประจำปีงบประมาณ 2562 ขอขอบคุณพี่ ๆ เพื่อน ๆ ในคณะที่ให้ความช่วยเหลือ ตลอดจนคำแนะนำที่เป็นประโยชน์ทั้งในการเรียนและการทำงานวิจัยนี้ ท้ายที่สุดขอขอบพระคุณบิดา มารดา ที่เป็นกำลังใจและให้โอกาสทางการศึกษาอันมีค่ายิ่งแก่ผู้วิจัย

ธนัท จรณะสมบูรณ์

# สารบัญ

หน้า

# สารบัญตาราง

**หน้า**

# สารบัญรูปภาพ

**หน้า**

# บทที่ 1
# บทนำ

## 1.1 ที่มาและความสำคัญของปัญหา

ธุรกิจร้านค้ามักจะเสนอโปรโมชั่นเพื่อดึงดูดลูกค้าใหม่ให้ซื้อสินค้า ซึ่งลูกค้าที่จัดได้ว่าเป็นลูกค้าที่มีคุณค่าต่อธุรกิจคือ ลูกค้าที่กลับมาซื้อสินค้าภายหลัง ซึ่งในปัจจุบันธุรกิจร้านค้าจำนวนมากมีการจัดโปรโมชั่นเสนอให้กับลูกค้า ซึ่งมีค่าใช้จ่ายสูงแต่อาจได้ลูกค้าที่มาซื้อสินค้าเพียงครั้งเดียวสำหรับโปรโมชั่นนี้เท่านั้น (one-time deal hunter) ซึ่งไม่เกิดประโยชน์ทางธุรกิจในระยะยาวต่อร้านค้า จากปัญหาดังกล่าวนี้ ผู้วิจัยได้นำประโยชน์ของข้อมูลประวัติธุรกรรมการซื้อสินค้า (shopping history) เพื่อทำนายการกลับมาซื้อซ้ำของลูกค้า (repeat buyer) หรือเรียกลูกค้ากลุ่มนี้ว่า ลูกค้าที่มีความภักดี (loyal customer)

ดังนั้นการทำนายว่าลูกค้าคนใดมีโอกาสกลับมาซื้อสินค้าซ้ำ (repeat buyer) จะช่วยในการคัดเลือกกลุ่มเป้าหมายในการจัดทำโปรโมชั่นและลดงบประมาณในส่วนของโปรโมชั่นที่ส่งให้กับลูกค้าไม่ก่อให้เกิดผลกำไรทางธุรกิจ จึงเป็นที่มาของงานวิจัยนี้ที่ผู้วิจัยนำเทคนิคการเรียนรู้ของเครื่องมาพัฒนาระบบการทำนายการซื้อซ้ำของผู้ซื้อ

ในงานวิจัยนี้ใช้ชุดข้อมูลจาก Kaggle “Acquire Valued Shopper Challenge” ซึ่งเป็นข้อมูลที่ใช้ในการแข่งขัน (Competition) การทำนายการซื้อซ้ำและเป็นชุดข้อมูลที่เปิดเผยบนเว็บไซต์ Kaggle [1] โดยชุดข้อมูลดังกล่าวประกอบด้วยข้อมูลประวัติการซื้อก่อนได้รับการเสนอโปรโมชั่นของกลุ่มลูกค้าจำนวนหนึ่ง ข้อมูลการเสนอโปรโมชั่นให้กับลูกค้าและผลลัพธ์การจำแนกลูกค้าว่าเป็นลูกค้าที่กลับมาซื้อซ้ำหรือไม่ ชุดข้อมูลนี้เป็นฐานข้อมูลขนาดประมาณ 20.7 GB ที่มีจำนวนข้อมูลประมาณ 350 ล้านรายการที่มีข้อมูลของลูกค้าที่ไม่มีการระบุตัวตนของผู้ซื้อ (Anonymized transactional data)

## 1.2 วัตถุประสงค์ของการวิจัย

1. เพื่อศึกษาการนำเทคนิคการเรียนรู้ของเครื่องไปใช้กับปัญหาการทำนายการซื้อซ้ำของผู้ซื้อโดยใช้เทคนิคการเรียนรู้ของเครื่อง
2. เพื่อศึกษาและเปรียบเทียบเทคนิคการทำ Feature Engineering และ Machine Learning วิธีต่าง ๆ ว่ามีผลต่อการแม่นยำในการทำนายอย่างไร

## 1.3 ขอบเขตของการวิจัย

งานวิจัยนี้พัฒนาแบบจำลองการทำนายโดยใช้เทคนิคการเรียนรู้ของเครื่อง (Machine Learning) ที่ทำงานในรูปแบบโปรแกรมคอมพิวเตอร์ โดยมีขอบเขตดังนี้

1. ใช้ข้อมูลจากเว็บไซต์ Kaggle ในการแข่งขัน “Acquire Valued Shoppers Challenge” โดยผลลัพธ์การทำนายแสดงถึงค่าความน่าจะเป็น (Probability) ของการกลับมาซื้อซ้ำ ซึ่งมีค่าระหว่าง 0-1
2. ศึกษาวิธีการทำ Feature Engineering
3. เปรียบเทียบเทคนิคการทำนายที่หลากหลายได้แก่ Random forest regressor, Random forest classifier, Gradient Boosting และ XGBoost
4. ประเมินประสิทธิภาพแบบจำลองการทำนายด้วยค่า Area Under Receiver Operating Characteristic Curve (AUC) ซึ่งเป็นไปตามกติกาการแข่งขันของ Kaggle

## 1.4 วิธีดำเนินการวิจัย

ขั้นตอนการดำเนินงานของงานวิจัยนี้มีดังนี้

1. ทบทวนวรรณกรรมและงานวิจัย (Literature Review) ที่เกี่ยวข้อง
2. พัฒนาโปรแกรมสำหรับการทำ Feature Engineering และการทำนายโดยใช้เทคนิคการเรียนรู้เครื่องได้แก่ Random forest regressor, Random forest classifier, Gradient Boosting และ XGBoost
3. ประเมินประสิทธิภาพของแบบจำลองการทำนายในรูปแบบของ AUC แล้วนำมาปรับปรุงโปรแกรมเพื่อให้แบบจำลองการทำนายมีความถูกต้องมากที่สุด
4. ประเมินและสรุปงานวิจัย

## 1.5 ประโยชน์ที่คาดว่าจะได้รับจากการวิจัย

ทราบถึงหลักการทำงานของ Random forest regressor, Random forest classifier, Gradient Boosting และ XGBoost เพื่อนำมาใช้งานได้อย่างเหมาะสม และได้ฟีเจอร์ที่เหมาะสมมาใช้ในการจำแนกผู้ซื้อ และทราบถึงปัจจัยที่ส่งผลกระทบต่อการตัดสินใจซื้อสินค้าของผู้ซื้อ เพื่อเพิ่มประสิทธิภาพในการจัดทำแผนการตลาดแบบสร้างแรงจูงใจในการซื้อให้ตรงกับกลุ่มเป้าหมาย เพื่อลดต้นทุนในการจัดทำแผนการตลาด

# บทที่ 2
# แนวคิด ทฤษฎี เทคโนโลยี และระบบงานที่เกี่ยวข้อง

การทำนาย (Prediction) เป็นส่วนหนึ่งของการเรียนรู้ของเครื่อง (Machine Learning) ซึ่งแบ่งออกเป็น 2 รูปแบบหลักๆ คือ (1) การทำนายด้วยการวิเคราะห์การถดถอย (Regression) เป็นการศึกษาความสัมพันธ์ระหว่างฟีเจอร์ และนำข้อมูลความสัมพันธ์ที่ได้จากการวิเคราะห์ไปทำนาย (2) การทำนายการด้วยการจำแนกข้อมูล (Classification) เป็นกระบวนการสร้างแบบจำลองการทำนายจากข้อมูลที่มีอยู่ โดยสร้างกฎหรือเงื่อนไขเพื่อช่วยในการตัดสินใจ

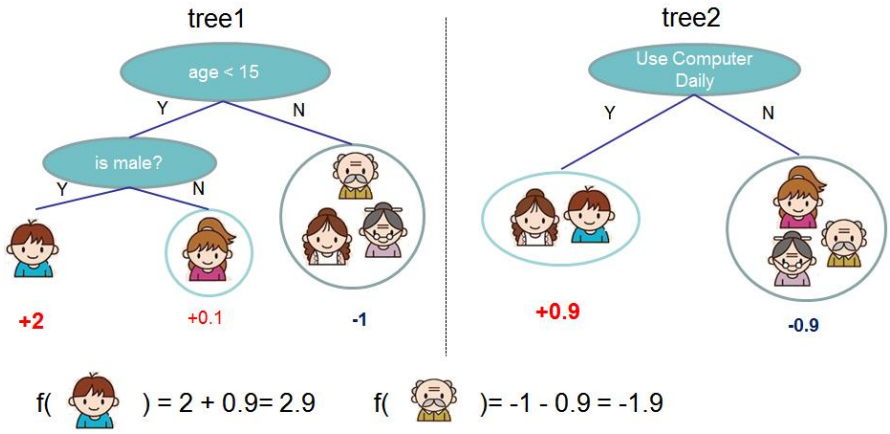
ในงานวิจัยนี้ผู้วิจัยได้ศึกษาเอกสารและงานวิจัยที่เกี่ยวข้องดังนี้

1. อัลกอริทึมในการสร้างแบบจำลองการทำนาย ได้แก่ Decision Tree, Random Forest Regressor, Random Forest Classifier, Gradient Boost, XGBoost
2. เทคนิคในการสร้างฟีเจอร์ (Feature Engineering) ได้แก่ Content-Based และ Collaborative filtering
3. ตัวชี้วัดผลประสิทธิภาพของแบบจำลองการทำนาย ได้แก่ Area Under the receiver operating characteristic Curve, Cross-validation, Leave One Out และ Cross-validation
4. งานวิจัยที่เกี่ยวข้องกับการทำนายการซื้อซ้ำ

## 2.1 ทฤษฎีพื้นฐานเกี่ยวกับ Machine Learning

### 2.1.1 ทฤษฎีเกี่ยวกับ Decision Tree

ต้นไม้ตัดสินใจ (Decision Tree) [2] เป็นเทคนิคที่ให้ผลลัพธ์ในลักษณะของโครงสร้างต้นไม้ เมื่อมีข้อมูลที่ต้องการจัดกลุ่มจะนำฟีเจอร์ต่าง ๆ ของข้อมูลไปเปรียบเทียบกับตามกิ่งของต้นไม้ (branch) จนกระทั่งถึงใบล่างสุด (leaf) ซึ่งเป็นกลุ่มของข้อมูลเหมือนกัน โดยภายในต้นไม้จะประกอบด้วยโหนด (node) โดยแต่ละโหนดมีการทดสอบฟีเจอร์โดยจำแนกตามกิ่งของต้นไม้ (branch) แสดงถึงค่าที่เป็นไปได้ของฟีเจอร์ที่เลือกมาทดสอบ และใบ (leaf) แสดงถึงการตัดสินใจของกลุ่มของข้อมูล (class) ซึ่งเป็นผลลัพธ์ที่ได้จากการทำนาย




ภาพประกอบ 1 รูปแบบโครงสร้างของ Decision Tree

ที่มา: https://xgboost.readthedocs.io/en/latest/tutorials/model.html

### 2.1.2 ทฤษฎีเกี่ยวกับ Random Forest

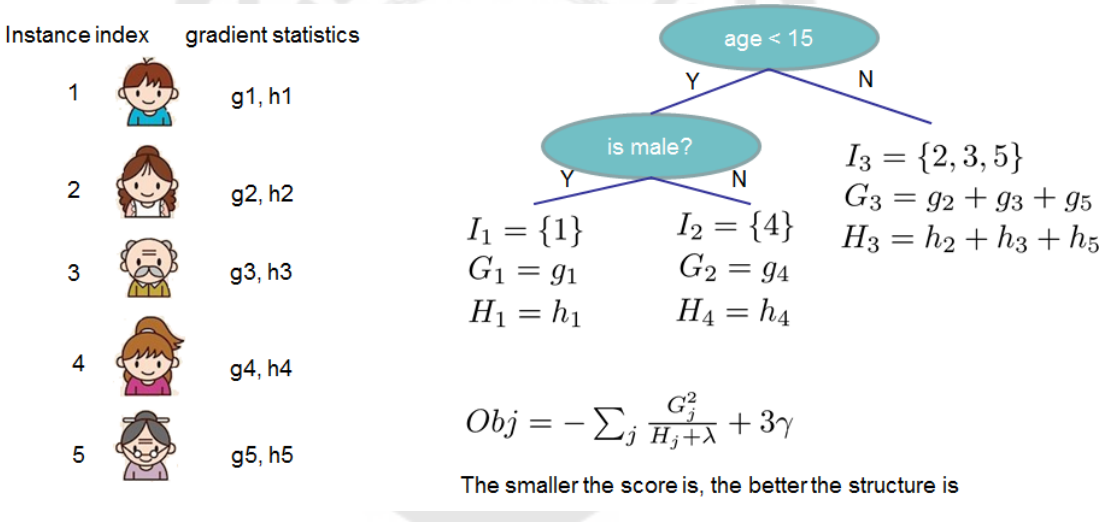
Random Forest (RF) [2] เป็นแบบจำลอง เป็นอีกการทำนายที่ใช้พื้นฐานจากต้นไม้ตัดสินใจ (Decision Tree) เป็นการทำนายแบบชุดของ Decision Tree หลายๆ ต้น (Ensemble of Decision Trees) โดยสร้างจากการสุ่มข้อมูลตัวอย่างแบบเลือกแล้วใส่กลับ (random sampling with replacement) เพื่อนำมาสร้างเป็นแบบจำลองต้นไม้โดยแต่ละต้นมีลักษณะที่ไม่ซ้ำกัน โดยแต่ละแบบจำลองจะมีการทำนายผล ซึ่งผลจากการทำนายของต้นไม้แต่ละต้นจะทำการโหวตเลือกผลการทำนายที่ได้รับการโหวตมากที่สุดวิธีการนี้เรียกว่า Bagging หรือ Boostraping โดยงานวิจัยนี้ใช้ Random forest ของ scikit-learn [3] (Random Forest Regressor) จะได้ผลลัพธ์ออกมาเป็นค่าความน่าจะเป็น สำหรับพารามิเตอร์หลักที่ปรับได้ในการสร้างแบบจำลอง Random Forest ได้แก่

- n_estimator: จำนวนต้นไม้ทั้งหมดในแบบจำลองการทำนาย ซึ่งโดยทั่วไปมีค่ายิ่งมากทำให้การทำนายแม่นยำ แต่จะใช้เวลานานในการคำนวณ
- max_features: จำนวนของฟีเจอร์ที่ถูกสุ่มมาสร้างต้นไม้ตัดสินใจ (Decision Tree) แต่ละต้น
- max_depth: จำนวนลำดับชั้นของต้นไม้ตัดสินใจ
- min_samples_leaf: จำนวนข้อมูลอย่างน้อยที่สุ่มมาใช้เป็น leaf node
- random_state: ค่าที่ใช้ในการสุ่มชุดข้อมูลในการฝึกสอนและชุดข้อมูลทดสอบ

### 2.1.3 ทฤษฎีเกี่ยวกับ Gradient Boosting Tree

Gradient Boosting Tree [2] เป็นการสร้างแบบจำลองที่ใช้เทคนิคต้นไม้ตัดสินใจ (Decision Tree) จำนวนหลายๆต้น (Ensemble of Decision Tree) มาช่วยกันทำนายโดยแตกต่างจาก Random Forest ตรงที่ต้นไม้ตัดสินใจแต่ละต้นจะถูกสร้างตามลำดับ (Sequence) โดยมีต้นไม้ตัดสินใจแต่ละตัวจะถูกสร้างขึ้นมาเพื่อแก้ไขผลการทำนายที่ผิดพลาดของต้นไม้ตัดสินใจก่อนหน้าซึ่งจะเหมือนกับ Random Forest Regressor สำหรับการสร้างแบบจำลองการทำนาย Gradient Boosting Tree มีพารามิเตอร์หลักที่ปรับได้ในการสร้างแบบจำลองได้แก่

- n_estimator: จำนวนต้นไม้ทั้งหมดในแบบจำลองการทำนาย
- max_depth: จำนวนลำดับชั้นของต้นไม้ตัดสินใจ
- learning_rate: ค่าที่ใช้ควบคุมความเข้มข้นที่ต้นไม้แต่ละต้นเข้าไปแก้ไขความผิดพลาดของต้นไม้ก่อนหน้า




ภาพประกอบ 2 โครงสร้างของ Gradient Boosting

ที่มา: https://xgboost.readthedocs.io/en/latest/tutorials/model.html

โดยในงานวิจัยนี้เลือกใช้ Gradient Boosting Regressor ของ scikit-learn [3] และไลบรารี XGBoost [4] ซึ่งเป็นไลบรารีสำหรับเทคนิค Gradient Boosting Tree ที่พัฒนาโดย Tianqi Chen ซึ่งเพิ่มทำให้เร็วในการคำนวณและเพิ่มประสิทธิภาพของแบบจำลอง

**2.1.4 ทฤษฎีเกี่ยวกับ Cross-validation**

Cross-validation [3] เป็นวิธีที่ได้รับความนิยมในการวัดประสิทธิภาพของแบบจำลองการทำนายเนื่องจากผลที่ได้มีความน่าเชื่อถือ การวัดประสิทธิภาพด้วยวิธี Cross-validation นี้จะทำการแบ่งข้อมูลออกเป็นหลายส่วนเท่า ๆ กัน เรียกว่า K-fold เช่น K-fold เท่ากับ 5 คือ การแบ่งข้อมูลออกเป็น 5 ส่วนโดยที่แต่ละส่วนจะมีจำนวนเท่ากัน จากนั้นนำข้อมูลทีละชุดมาใช้เป็นตัววัดประสิทธิภาพของแบบจำลองการทำนาย และใช้ข้อมูลที่เหลือเป็นข้อมูลในการเทรน ทำวนไปเรื่อย ๆ จนครบจำนวนที่แบ่งไว้

**2.1.5 ทฤษฎีเกี่ยวกับ Leave One Out Cross Validation**

Leave One Out [5] เป็นวิธีการตรวจสอบความถูกต้องโดยการทดสอบข้อมูลทั้งหมดเท่ากับจำนวนข้อมูล N ตัว โดยการนำข้อมูลเทรนออกมาทีละหนึ่งตัวเพื่อใช้เป็นข้อมูลทดสอบ และนำข้อมูลที่เหลือเข้าใช้เทรน แบบจำลองการทำนายทำแบบนี้วนไปเรื่อย ๆ จนครบทุกตัว

**2.1.6 ทฤษฎีเกี่ยวกับ Content-based**

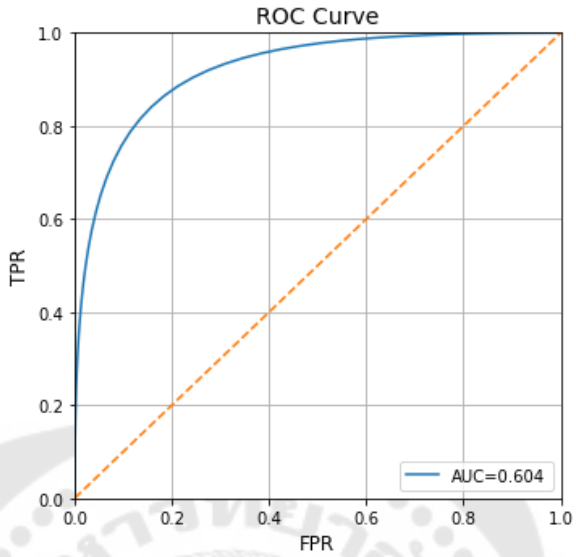
Content-based [6] เป็นเทคนิคการแนะนำโดยเลือกพิจารณาจากลักษณะของสินค้าที่จะแนะนำว่ามีความคล้ายคลึงกับคุณลักษณะของสินค้าที่ปัจจุบันผู้ซื้อเคยซื้อหรือใช้มาก่อน และมีคะแนนความชอบค่อนข้างสูง วิธีการนี้ต้องใช้ข้อมูลอธิบายลักษณะของสินค้าได้ เพื่อช่วยในการหาลักษณะความชอบของสินค้านั้น โดยในงานวิจัยนี้ผู้วิจัยได้ใช้เทคนิคนี้ในการสร้างฟีเจอร์

**2.1.7 ทฤษฎีเกี่ยวกับ Collaborative filtering**

Collaborative filtering [6] การแนะนำโดยเลือกพิจารณาจากลักษณะของลูกค้าที่เคยกลับมาซื้อสินค้าไปแล้วว่ามีความคล้ายคลึงกับลูกค้าใหม่และแนะนำสินค้าที่คล้ายคลึงกันให้กับลูกค้าเพื่อจะได้มีโอกาสมากกว่าที่ลูกค้าจะชอบสินค้าคล้าย ๆ กัน

**2.1.8 ทฤษฎีเกี่ยวกับ Area Under the receiver operating characteristic Curve (AUC)**

Receiver Operating Characteristics (ROC) เป็นตัววัดประสิทธิภาพโมเดลการจำแนกประเภทโดยแสดงกราฟความสัมพันธ์ระหว่างข้อมูลที่ทำนายถูก (แกน Y: True Positive rate) และข้อมูลที่ทำนายผิด (แกน X: False Positive rate) โดยถ้ามีค่าเข้าใกล้ 1 แสดงว่ามีประสิทธิภาพดี เนื่องจากมีค่าที่ทำนายถูก (True Positive) โดยค่า Area Under Curve (AUC) ใช้แสดงพื้นที่ใต้กราฟ ROC โดยมีค่าเริ่มต้นที่ 0 ถึง 1 โดยที่ 0 หมายถึงแบบจำลองนั้นมีประสิทธิภาพต่ำกว่า และ 1 หมายถึงแบบจำลองนั้นมีประสิทธิภาพสูงที่สุดตามภาพประกอบ 3 โดยในงานวิจัยนี้จะใช้ค่า AUC การส่งผลให้กับ Kaggle เพื่อวัดค่าความถูกต้อง




ภาพประกอบ 3 ตัวอย่าง ROC Curve

ที่มา: https://towardsdatascience.com/receiver-operating-characteristic-curves-demystified-in-python-bd531a4364d0

## 2.2 งานวิจัยที่เกี่ยวข้อง

การทบทวนวรรณกรรมของงานวิจัยนี้ ได้ทำการศึกษางานวิจัยที่เกี่ยวข้องกับการทำนายการซื้อซ้ำของผู้ซื้อโดยเน้นการเรียนรู้ของเครื่องด้วยเทคนิคต่าง ๆ มีรายละเอียดดังต่อไปนี้

(1) บทความวิจัยเรื่อง “Prediction of the shoppers loyalty with aggregated data streams” [7]

ในงานวิจัยนี้เสนอวิธีการสร้างแบบจำลองการทำนายการซื้อซ้ำของลูกค้าที่จะกลับมาซื้อหลังจากได้ส่วนลดไปแล้ว ด้วยข้อมูลประวัติการซื้อสินค้าของผู้ซื้อโดยใช้ชุดข้อมูลจากเว็บไซต์ “Kaggle” ในการแข่งขัน “Kaggle Acquire Valued Shopper Challenge” [1] ในปี 2014 ด้วยการทำ Feature Engineering โดยใช้อัลกอริทึม Random Forest, gbR, glmNet, Vowpal Wabbit และ nnet แบบเพื่อเปรียบเทียบผลการทำนาย

โดยเปรียบเทียบความแม่นยำด้วยค่า AUC ซึ่งสรุปได้ว่า Random Forest นั้นให้ค่าสูงที่สุด โดยความถูกต้องของการทำนายว่าลูกค้าคนใด ที่จะเป็นลูกค้าที่กลับมาซื้อหลังจากได้รับข้อเสนอ

ไปแล้ว เป็นแบบ Area under receiver operating curve (AUC) ได้ผลลัพธ์ 0.61172 และได้ที่ 13 สำหรับการแข่งขันนี้

(2) บทความวิจัยเรื่อง “Generic Framework to Predict Repeat Behavior of Customers Using Their Transaction History” [8]

งานวิจัยนี้มีบทบาทสำคัญต่อการกำหนดงบประมาณในการทำโฆษณาการจัดวางผลิตภัณฑ์และการกำหนดเป้าหมายลูกค้าที่เกี่ยวข้อง แก้ปัญหานี้ด้วย Standard Predictive Model โดยนำเสนอ meta-model ที่จะใช้กับ Dimension ที่แตกต่างกันที่มีอยู่ในชุดข้อมูล Transaction ซึ่ง Dimension สามารถเป็นได้ทั้ง customer, product, offer, target, marketplace และ transactions

Framework ของงานวิจัยนี้ได้แก่ การสร้าง feature set ที่ครอบคลุมรวมถึงอัลกอริทึมของแบบจำลองการทำนายเพื่อเรียนรู้ Predictive model และ Common Data Format (CDF) คือหมวดหมู่กว้างๆ ของข้อมูล transaction ใช้เพื่อระบุความแตกต่างของ dimension ของข้อมูล Transaction แบบกว้างๆ โดย Feature Engineering ทำงานร่วมกับ user data และ CDF จนถึงการทำ Data Schema mapping เพื่อสร้าง feature set ที่ครบถ้วนสมบูรณ์ได้แก่ Customer based features, Target Entity-based features, Customer-Target Entity Interaction-based features, Product-based features, Similarity-based features

งานวิจัยนี้ได้อธิบายแนวคิดทั่วไปสำหรับการคาดการณ์พฤติกรรมการซื้อซ้ำของลูกค้าพร้อมกับโมดูลการสร้างคุณลักษณะที่เป็นนามธรรมซึ่งทำงานได้กับชุดข้อมูลหลายประเภท และยังอธิบายการเพิ่มประสิทธิภาพในการให้ข้อเสนอที่ควรทำกับลูกค้า สุดท้ายทำการทดลองกับชุดข้อมูล Kaggle “Acquired Valued Shopper” [9] และ IJCAI 2015 “Repeat Buyer Prediction” [10]

(3) บทความวิจัยเรื่อง “Deep Temporal Features to Predict Repeat Buyers” [11]

ในงานวิจัยนี้จะกล่าวถึงการแยกแยะ ผู้ซื้อที่จะกลับมาซื้อซ้ำหลังจากการซื้อที่ได้รับโปรโมชั่นในครั้งแรกไปแล้ว โดยมุ่งเน้นไปที่ประโยชน์จากผู้ซื้อที่ภักดีเพื่อเป็นเป้าหมายของแคมเปญทางการตลาด เพื่อช่วยในการลดต้นทุนในการทำโปรโมชั่นและเพิ่มผลตอบแทนจากการลงทุน และทำให้ลูกค้าได้รับข้อเสนอที่ตรงกับความต้องการ โดยนำประวัติธุรกรรมของลูกค้ามาสร้างแบบจำลองการทำนายพฤติกรรมการซื้อซ้ำของลูกค้าสำหรับผลิตภัณฑ์ต่าง ๆ โดยใช้เทคนิค Feature Extraction เพื่อสร้างคุณสมบัติสามประเภท ได้แก่ Customer-based features, Product-based features, และ Customer-Product interaction based features และใช้แบบจำลองการเรียนรู้ QR based aggregate level model, LSTM based temporal model, Mixture of Experts over QR and LSTM models

จากผลการทดลองกับข้อมูลจาก “Kaggle Acquire Valued Shopper Challenge” ข้อมูลประวัติธุรกรรมย้อนหลัง 1 ปีก่อนลูกค้าจะได้รับแรงจูงใจจากโปรโมชั่น สามารถแยกคุณสมบัติต่าง ๆ จากข้อมูลประวัติการทำธุรกรรมทำ Feature Extraction ได้สร้าง Feature จำนวน 88 รายการสำหรับ QR และ 19 temporal features สำหรับ LSTM สร้างโมเดลการตลาดโดยแบ่งออกเป็น 9 ตลาดจากลูกค้าทั้งหมด 38,000 คน 28.8% คือลูกค้าจาก 9 ตลาด

(4) โค้ดจาก Github ของ Auduno [9]

Auduno เป็นหนึ่งในผู้แข่งขันในงาน Kaggle Acquire Valued Shopper Challenge ที่ได้สร้าง Features เพิ่มเติมจากข้อมูลประวัติธุรกรรมการซื้อสินค้า ข้อมูลการได้รับข้อเสนอโปรโมชั่นของผู้ซื้อ และข้อมูลรายละเอียดของโปรโมชั่น ได้แก่ ส่วนแบ่งทางการตลาดของสินค้าในแต่ละหมวดหมู่ (market share of each product in category) ราคาสินค้าโดยเฉลี่ย (average price for the product) ส่วนแบ่งของลูกค้าที่ซื้อสินค้านี้หรือหมวดหมู่นี้ (share of customers who bought the product/category) การใช้จ่ายในแต่ละฤดูกาลของสินค้านั้น (Seasonal spending per product) อัตราการกลับมาซื้อซ้ำของสินค้าโดยดูจากประวัติการซื้อ (General repeat-buy-probabilities for a product base on historical repeat buys) จำนวนการแข่งขันของสินค้า (how much competition a product faces) ความคุ้มค่าของสินค้าเมื่อเปรียบเทียบกับสินค้าอื่น (how cheap a product is compared to other products) ระยะเวลาตั้งแต่การซื้อครั้งแรก (Time since customers first transaction) และใช้สร้างแบบจำลองการทำนายจากหลายอัลกอริทึมมารวมกัน ได้แก่ XGBoost, ExtraTreeClassifier, Sofia-ml และ Vowpal wabbit โดยการนำผลลัพธ์ของแต่ละแบบจำลองมาหาค่าเฉลี่ย

(5) Marios Michailidis [12]

Marios Michailidis คือผู้ชนะการแข่งขัน Kaggle Acquire Valued Shopper Challenge ได้สำรวจพบว่าข้อมูล train และข้อมูล test มีข้อมูลผู้ซื้อและข้อมูลโปรโมชั่นที่แตกต่างกันมาก เพื่อแก้ไขปัญหานี้ จึงนำเทคนิค Leave One Out มาใช้ในการเทรนแบบจำลองการทำนายโดยการเลือกข้อมูลออกมาทีละโปรโมชั่น (Offer) เพื่อใช้เป็นข้อมูล test และใช้ข้อมูลที่เหลือเป็นข้อมูล train โดยเรียกเทคนิคนี้ว่า “Leave One Offer Out” และสร้างฟีเจอร์ด้วยการใช้หลัก Content-Based และ Collaborative filtering

# บทที่ 3
# การพัฒนาระบบการทำนายการซื้อซ้ำของผู้ซื้อโดยใช้เทคนิคการเรียนรู้ของเครื่อง

ในบทนี้ผู้วิจัยได้นำเสนอการพัฒนาระบบการทำนายการซื้อซ้ำของลูกค้าจากประวัติธุรกรรมในชุดข้อมูลของ “Kaggle Acquire Valued Shopper competition” [1] มาใช้ในงานวิจัยโดยใช้เทคนิคการเรียนรู้ของเครื่อง โดยเริ่มจากศึกษาชุดข้อมูล การสร้างฟีเจอร์ (feature engineering) และสร้างแบบจำลองการทำนายแบบด้วย Random Forest Regressor, Random Forest Classifier, Gradient Boost และ XGBoost โดยมีรายละเอียดดังนี้

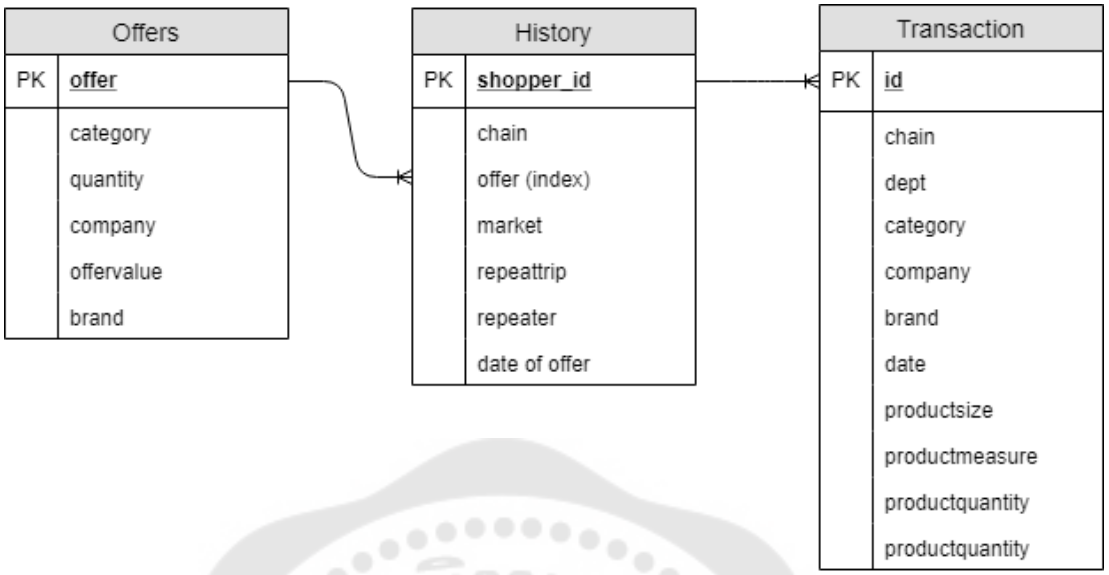
## 3.1 นิยามของปัญหา “Acquire Valued Shopper Challenge”

การจำแนกลูกค้าที่มีความภักดี (loyal customer) ที่มีโอกาสกลับมาซื้อสินค้าเมื่อมีการเสนอโปรโมชั่นหนึ่งๆ หรือจะเป็นลูกค้าที่มาซื้อเพียงครั้งเดียวเพื่อรับโปรโมชั่น (one-time deal hunters) จากประวัติการซื้อสินค้า

## 3.2 อธิบายชุดข้อมูล (Data Exploration)

ชุดข้อมูลนี้เป็นชุดข้อมูลเกี่ยวกับประวัติธุรกรรมการซื้อสินค้า รายละเอียดข้อมูลการจัดโปรโมชั่นที่เคยจัดขึ้น รวมถึงประวัติการได้รับโปรโมชั่นของลูกค้า โดยมีข้อมูลทั้งหมด 349,655,789 รายการที่ไม่ระบุตัวตนผู้ซื้อจากผู้ซื้อกว่า 300,000 คน และ 160,057 ผู้ซื้อในชุดข้อมูลฝึกสอน (training set) ในกลุ่มนี้มีลูกค้าที่ภักดี (repeat buyer) อยู่ 43438 คนหรือประมาณ 27.14% และ 151,484 คนในชุดข้อมูลทดสอบ (test set) โดยชุดฝึกสอน (training set) จะมี Label 0 และ 1 เพื่อบอกว่าลูกค้าคนใดเป็นลูกค้าที่กลับมาซื้อซ้ำ โดยมีโครงสร้างความสัมพันธ์ของข้อมูลตามภาพประกอบ 4 และข้อมูลทั้งหมดจะแบ่งออกเป็น 3 ตารางได้แก่

1. ข้อมูลประวัติธุรกรรมการซื้อสินค้าตามตาราง 1
2. ข้อมูลรายละเอียดข้อเสนอโปรโมชั่นตามตาราง 2
3. ประวัติการได้รับข้อเสนอโปรโมชั่นของผู้ซื้อตามตาราง 3




ภาพประกอบ 4 ER-diagram

ตาราง 1 Transaction

| | |
|---|---|
| id | รหัสประจำตัวลูกค้า (ไม่ซ้ำ) |
| chain | รหัสสาขา |
| dept | กลุ่มของหมวดหมู่ |
| category | หมวดหมู่ของสินค้า |
| company | จำนวนครั้งที่ลูกค้าซื้อซ้ำ |
| brand | ค่า “จริง” หรือ “ไม่จริง” โดยดูจากจำนวนครั้งที่ซื้อซ้ำ ถ้าเคยซื้อซ้ำจะถือว่าเป็นจริง |
| date | วันที่ลูกค้าได้รับข้อเสนอ |
| Product size | จำนวนของสินค้าที่ซื้อ (เช่น น้ำ 16 ออนซ์) |
| Product measure | หน่วยของสินค้า (เช่น ออนซ์) |
| Purchase quantity | จำนวนต่อหน่วยที่ซื้อ |
| Purchase amount | จำนวนเงินที่ซื้อ |

โดยมีตัวอย่างข้อมูลประวัติธุรกรรมการซื้อสินค้าแสดงดังตาราง 2

ตาราง 2 ตัวอย่างข้อมูลประวัติธุรกรรมการซื้อสินค้าในตาราง Transaction

| id | chain | dept | category | company | brand | date | Product size | Product measure | Purchase quantity | Purchase amount |
|---|---|---|---|---|---|---|---|---|---|---|
| 86246 | 205 | 7 | 707 | 1078778070 | 12564 | 3/2/2012 | 12 | OZ | 1 | 7.59 |
| 86252 | 205 | 99 | 9908 | 103338333 | 33170 | 5/14/2012 | 8 | OZ | 2 | 4.18 |
| 12332190 | 95 | 58 | 5834 | 103196131 | 15963 | 6/8/2013 | 12 | OZ | 1 | 5.99 |
| 12277270 | 95 | 36 | 3601 | 104900040 | 3809 | 8/4/2012 | 96 | OZ | 1 | 4 |
| 12262064 | 95 | 11 | 1102 | 104480040 | 22728 | 7/14/2012 | 100 | CT | 1 | 3.59 |

ตาราง 3 Offers

| | | |
|---|---|---|
| 0 | offer (index) | รหัสประจำตัวลูกค้า (ไม่ซ้ำ) |
| 1 | Category | รหัสสาขา |
| 2 | Quantity | จำนวนหน่วยที่ต้องซื้อเพื่อให้ได้รับส่วนลด |
| 3 | Company | จำนวนครั้งที่ลูกค้าซื้อซ้ำ |
| 4 | Offer value | จำนวนเงินที่ได้รับจากส่วนลด |
| 5 | Brand | รหัสยี่ห้อของสินค้า |

เก็บรวบรวมข้อมูลข้อเสนอต่าง ๆ ที่สร้างแรงจูงใจในการซื้อของลูกค้า เช่น คูปอง ส่วนลดต่าง ๆ โดยมีทั้งหมด 37 รายการ โดยมีตัวอย่างข้อมูลแสดงดังตาราง 4

ตาราง 4 ตัวอย่างข้อมูลจากตาราง Offers

| offer | category | quantity | company | offervalue | brand |
|---|---|---|---|---|---|
| 1190530 | 9115 | 1 | 108500080 | 5 | 93904 |
| 1194044 | 9909 | 1 | 107127979 | 1 | 6732 |
| 1197502 | 3203 | 1 | 106414464 | 0.75 | 13474 |
| 1198271 | 5558 | 1 | 107120272 | 1.5 | 5072 |
| 1198272 | 5558 | 1 | 107120272 | 1.5 | 5072 |

ตาราง 5 History

| | |
|---|---|
| 0 | Shopper ID |
| 1 | Chain |
| 2 | Offer (index) |
| 3 | Market |
| 4 | Repeat trips |
| 5 | Repeater |
| 6 | Date of offer |

ข้อมูลฝึกสอน (training data) และข้อมูลทดสอบ (test data) โดยฟิล์ดข้อมูลที่ 4 คือ จำนวนครั้งที่กลับมาซื้อสินค้า (repeat trips) และ 5 คือคำตอบว่าผู้ซื้อคนนั้นจะกลับมาซื้อซ้ำหรือไม่ (repeater) หรือเรียกว่า (label) จะไม่มีในชุดข้อมูลทดสอบโดยมีตัวอย่างข้อมูลแสดงดังตาราง 6

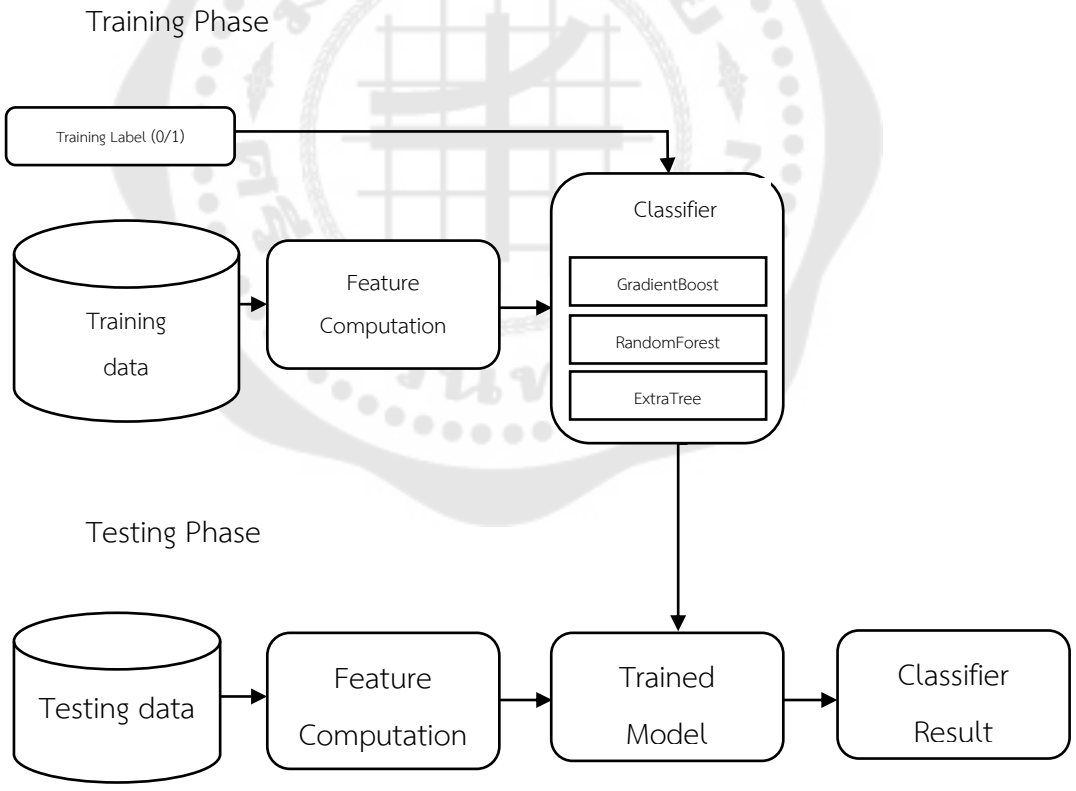
ตาราง 6 ตัวอย่างข้อมูลตาราง History

| id | chain | offer | market | repeattrips | repeater | offerdate |
|---|---|---|---|---|---|---|
| 86246 | 205 | 1208251 | 34 | 5 | t | 4/24/2013 |
| 86252 | 205 | 1197502 | 34 | 16 | t | 3/27/2013 |
| 12682470 | 18 | 1197502 | 11 | 0 | f | 3/28/2013 |
| 12996040 | 15 | 1197502 | 9 | 0 | f | 3/25/2013 |
| 13089312 | 15 | 1204821 | 9 | 0 | f | 4/1/2013 |

## 3.3 การพัฒนาระบบทำนายการซื้อซ้ำของผู้ซื้อโดยใช้เทคนิคการเรียนรู้ของเครื่อง

### 3.3.1 ภาพรวมของระบบ (System Overview)

ภาพรวมของระบบประกอบด้วย 2 ส่วนได้แก่ Training Phase และ Testing Phase โดยในส่วนของ Training Phase ได้ใช้ข้อมูลคำตอบจากตาราง History (repeater) 0 หรือ 1 โดย 1 หมายถึงลูกค้าที่จะกลับมาซื้อสินค้าซ้ำและ 0 หมายถึงลูกค้าที่มาซื้อสินค้าเพียงครั้งเดียวเพื่อใช้โปรโมชั่น เพื่อทำการสร้างฟีเจอร์ (Feature Engineering) จากการรวบรวมข้อมูลประวัติธุรกรรมและข้อมูลรายละเอียดโปรโมชั่น เพื่อลดขนาดของข้อมูลที่จะใช้ในการฝึกสอน (Train) แบบจำลองการทำนายโดยใช้ 4 อัลกอริทึมในการสร้างแบบจำลองได้แก่ Random forest regressor, Random forest classifier, Gradient Boost และ XGBoost ใน Testing Phase จะสร้างฟีเจอร์ตาม Training Phase แต่ใช้แบบจำลองการทำนายที่ได้จาก Training Phase ในการสร้างผลลัพธ์การทำนายตามภาพประกอบ 5




ภาพประกอบ 5 ภาพรวมของระบบ (System Overview)

### 3.3.2 รายละเอียดของระบบ

(1) การเตรียมข้อมูล (Data Preparation)

จากการสำรวจข้อมูลโปรโมชั่น (offer) พบว่าบริษัท (company), หมวดหมู่ (category) หรือ แบรนด์ (brand) ในประวัติการได้รับโปรโมชั่นของลูกค้า (ตาราง History) ไม่ได้ครอบคลุมข้อมูลประวัติการซื้อสินค้าทั้งหมด ผู้วิจัยจึงเลือกเอาข้อมูลที่ บริษัท หมวดหมู่ หรือแบรนด์ไม่อยู่ในชุดข้อมูลโปรโมชั่นออกจากประวัติการซื้อสินค้าทั้งหมด (transaction) เพื่อลดขนาดของข้อมูลที่ใช้ในการทำนาย เนื่องจากข้อมูลดังกล่าวไม่ได้ใช้กับการจัดโปรโมชั่นที่จะใช้ในทำนาย เพื่อลดขนาดของข้อมูลเหลือ 1.66 GB จากทั้งหมด 20.7 GB

โดยส่วนย่อยของโค้ด (code snippet) แสดงได้ดังนี้

```
1.  def reduce_data(offers_file, transactions_file, reduced_file):
2.      start = datetime.now()
3.      #get all categories and comps on offer in a dict
4.      offers_cat = {}
5.      offers_co = {}
6.      for e, line in enumerate( open(offers_file) ):
7.          offers_cat[ line.split(",")[1] ] = 1
8.          offers_co[ line.split(",")[3] ] = 1
9.      #open output file
10.     with open(reduced_file, "wb") as outfile:
11.         #go through transactions file and reduce
12.         reduced = 0
13.         for e, line in enumerate( open(transactions_file) ):
14.             if e == 0:
15.                 outfile.write( line ) #print header
16.             else:
17.                 #only write when if category in offers dict
18.                 if line.split(",")[3] in offers_cat or line.split(",")[4] in
    offers_co:
19.                     outfile.write( line )
20.                     reduced += 1
```

(2) การสร้าง Features (Feature Engineering)

จากการศึกษางานวิจัยที่เกี่ยวข้องของ Nikulin [7] และ Michailidis [12] ได้สร้างฟีเจอร์โดยใช้หลักการ Content-based [11] และ Collaborative filtering [11] คือการสร้างฟีเจอร์โดยดูจากลักษณะของสินค้าที่ผู้ซื้อสนใจร่วมกับดูลักษณะของผู้ซื้อที่คล้ายคลึงกัน เช่น ผู้ซื้อเคยมีการซื้อสินค้าจากบริษัท หมวดหมู่ แบรนด์เดิมก็จะมีโอกาสที่จะซื้อสินค้าสูงขึ้นเมื่อได้รับโปรโมชั่นจากสินค้าของบริษัท หมวดหมู่ หรือแบรนด์นั้น และกลับมาซื้ออีกครั้ง โดยฟีเจอร์ที่มีความสำคัญต่อการตัดสินใจได้แก่ Company, Category, Brand สามารถหาความนิยมของบริษัท หมวดหมู่ หรือแบรนด์ โดยดูจากจำนวนลูกค้าที่ชื่นชอบสินค้าดังกล่าวจากข้อมูลประวัติการซื้อสินค้าและข้อมูลโปรโมชั่น และผู้วิจัยได้หาความสำคัญของแต่ละฟีเจอร์ (feature importance) โดยใช้ Random Forest ของ scikit-learn ในการคำนวณหา Features Importance พบว่าฟีเจอร์ที่มีความสำคัญสูงสุด 3 ลำดับแรก ได้แก่ “Company”, “Category” และ “Brand” ดังตาราง 11 ซึ่งได้ผลลัพธ์เหมือนกับงานวิจัยของ Nikulin และเนื่องจากในชุดข้อมูลที่ได้มาไม่มีข้อมูลที่ระบุถึงสินค้าแน่นอน ผู้วิจัยจึงเลือก 3 ฟีเจอร์ที่กล่าวมารวมกันเรียกว่า “Product”

มีรายละเอียดการสร้างฟีเจอร์โดยการรวมข้อมูล (Feature Aggregation) ดังนี้

1. จำนวนครั้งที่ลูกค้าคนหนึ่งเคยซื้อจากบริษัทหนึ่ง
2. จำนวนเงินที่ลูกค้าคนหนึ่งเคยซื้อจากบริษัทหนึ่ง
3. จำนวนสินค้าที่ลูกค้าคนหนึ่งที่เคยซื้อจากบริษัทหนึ่ง
4. จำนวนครั้งที่ลูกค้าซื้อจากบริษัทหนึ่งภายใน 30 วันก่อนที่จะได้รับโปรโมชั่น
5. จำนวนครั้งที่ลูกค้าซื้อจากบริษัทหนึ่งภายใน 60 วันก่อนที่จะได้รับโปรโมชั่น
6. จำนวนครั้งที่ลูกค้าซื้อจากบริษัทหนึ่งภายใน 90 วันก่อนที่จะได้รับโปรโมชั่น
7. จำนวนครั้งที่ลูกค้าซื้อจากบริษัทหนึ่งภายใน 120 วันก่อนที่จะได้รับโปรโมชั่น
8. จำนวนครั้งที่ลูกค้าซื้อจากบริษัทหนึ่งภายใน 150 วันก่อนที่จะได้รับโปรโมชั่น
9. จำนวนครั้งที่ลูกค้าซื้อจากบริษัทหนึ่งภายใน 180 วันก่อนที่จะได้รับโปรโมชั่น
10. ไม่เคยซื้อจากบริษัทนี้เลย เป็น negative feature บ่งบอกว่าลูกค้าไม่เคยซื้อของจากบริษัทนี้มาก่อน

โดยฟีเจอร์ด้านบนสร้างโดยให้ความสำคัญกับฟีเจอร์ “บริษัท” เท่านั้นและพิจารณาในส่วนของ “แบรนด์” และ “หมวดหมู่” ในทำนองเดียวกัน จากการทดลองผู้วิจัยใช้ช่วงเวลา 6 เดือน คือ 30, 60, 90, 120 และ 180 วัน โดยจะเรียก Feature ชุดนี้ว่า “Base features” ตัวอย่าง base features ตามตาราง 7

ตาราง 7 ตัวอย่าง base features

| Label | Repeat trips | Id | Offer id | Never bought company | Never bought category | Never bought brand | Has bought brand company |
|---|---|---|---|---|---|---|---|
| 1 | 5 | 86246 | 1208251 | 0 | 1 | 0 | 0 |
| 1 | 16 | 86252 | 1197502 | 0 | 0 | 0 | 1 |
| 0 | 0 | 12682470 | 1197502 | 1 | 0 | 1 | 0 |
| 0 | 0 | 12996040 | 1197502 | 1 | 1 | 1 | 0 |
| 0 | 0 | 13089312 | 1204821 | 0 | 1 | 0 | 0 |

และสร้าง features เพิ่มเติมโดยศึกษาจากโค้ดของ Auduno [9] ได้แก่

1. ส่วนแบ่งทางการตลาดของสินค้าในแต่ละหมวดหมู่ (market share of each product in category)
2. ราคาสินค้าโดยเฉลี่ย (average price for the product)
3. ส่วนแบ่งของลูกค้าที่ซื้อสินค้านี้หรือหมวดหมู่นี้ (share of customers who bought the product/category)
4. การใช้จ่ายในแต่ละฤดูกาลของสินค้านั้น (Seasonal spending per product)
5. อัตราการกลับมาซื้อซ้ำของสินค้าโดยดูจากประวัติการซื้อ (General repeat-buy-probabilities for a product base on historical repeat buys)
6. จำนวนการแข่งขันของสินค้า (how much competition a product faces)
7. ความคุ้มค่าของสินค้าเมื่อเปรียบเทียบกับสินค้าอื่น (how cheap a product is compared to other products)
8. ระยะเวลาตั้งแต่การซื้อครั้งแรก (Time since customers first transaction)

โดยเรียกข้อมูล Features ทั้งหมดนี้รวมกับ Base features ว่า “All features”

### 3.3.3 อัลกอริทึมและแบบจำลองการทำนาย (Model Training)

ในส่วนนี้ผู้วิจัยใช้เทคนิคการเรียนรู้ของเครื่อง (Machine learning techniques) สำหรับสร้างแบบจำลองการทำนายการกลับมาซื้อซ้ำของผู้ซื้อโดยใช้ Random forest regressor, Random forest classifier, XGBoost และ Gradient Boot และปรับ tune พารามิเตอร์ที่เหมาะสมร่วมกับเทคนิค Leave One Offer Out [12]

ในงานวิจัยนี้ผู้วิจัยทำนายความน่าจะเป็นที่ลูกค้าจะกลับมาซื้อสินค้าซ้ำหลังจากได้รับโปรโมชั่นในรูปแบบ 0 ถึง 1 โดยใช้แบบจำลองประเภทต่าง ๆ และประเมินประสิทธิภาพของแบบจำลองด้วย Area Under the receiver operating characteristic Curve (AUC) ดังนี้

*3.3.3.1 สร้างแบบจำลองการทำนายโดยใช้ Random Forest Regressor ของ scikit-learn*

โดยส่วนย่อยของโค้ด (code snippet) แสดงได้ดังนี้

```
clf = RandomForestRegressor()
clf.fit(X_train, y_train)
predRF = clf.predict(X_test)
```

*3.3.3.2 สร้างแบบจำลองการทำนายโดยใช้ Random forest classifier ของ scikit-learn*

โดยส่วนย่อยของโค้ด (code snippet) แสดงได้ดังนี้

```
clf = RandomForestClassifier()
clf.fit(X_train, y_train)
predRF = clf.predict(X_test)
```

*3.3.3.3 สร้างแบบจำลองการทำนายโดยใช้ Gradient Boosting Regressor ของ scikit-learn*

โดยส่วนย่อยของโค้ด (code snippet) แสดงได้ดังนี้

```
clf = GradientBoostingRegressor()
clf.fit(X_train, y_train)
predRF = clf.predict(X_test)
```

*3.3.3.4 สร้างแบบจำลองการทำนายโดยใช้ XGBoost ของ scikit-learn*

โดยส่วนย่อยของโค้ด (code snippet) แสดงได้ดังนี้

```
param['eval_metric'] = 'auc'
dtrain = xgb.DMatrix(X_train, label=y_train)
clf = xgb.train(param, dtrain)
dtest = xgb.DMatrix(X_test)
predRF = clf.predict(dtest)
```

เมื่อรันโปรแกรมแล้วจะได้ผลการทำนายความน่าจะเป็นที่ผู้ซื้อมีโอกาสจะกลับมาซื้อซ้ำ ตัวอย่างตามตาราง 8 ซึ่งผลลัพธ์นี้ใช้ในการส่งในการแข่งขันของ Kaggle ผลลัพธ์ของการส่งในการแข่งขันแสดงดังตาราง 12 ซึ่งจะกล่าวถึงรายละเอียดต่อไป

ตาราง 8 ตัวอย่างผลลัพธ์ของแบบจำลองการทำนายของผู้ซื้อแต่ละคนจำแนกตามอัลกอริทึม

| | Repeat Probability | | | |
|---|---|---|---|---|
| Id | Random forest regressor | Random forest classifier | Gradient boosting regressor | XGBoost |
| 12262064 | 0.238325 | 0.34416064 | 0.34416064 | 0.44184661 |
| 12277270 | 0.325746 | 0.63062609 | 0.63062609 | 0.47733018 |
| 12332190 | 0.281708 | 0.07709697 | 0.07709697 | 0.20715557 |
| 12524696 | 0.122981 | 0.22649302 | 0.22649302 | 0.22845522 |
| 13074629 | 0.166164 | 0.25424540 | 0.25424540 | 0.24076188 |

*3.3.3.5 Public Score และ Private Score*

ในการแข่งขัน Acquire Valued Shoppers Challenge จะวัดประสิทธิภาพการทำนายด้วยคะแนน 2 ประเภทได้แก่

1. Public Score คำนวณจาก 20% ของข้อมูล test และอีก 80% ที่เหลือคำนวณจากข้อมูลอื่น
2. Private Score คำนวณจากข้อมูล test 80 % และอีก 20% ที่เหลือจะคำนวณจากข้อมูลอื่น

*3.3.3.6 Leave One Out Cross-validation*

ผู้วิจัยพบว่าข้อมูลประวัติการได้รับโปรโมชั่นของลูกค้าในชุดข้อมูลฝึกสอน (Train set) และข้อมูลทดสอบ (Test set) มีข้อมูลผู้ซื้อและโปรโมชั่นที่แตกต่างกันค่อนข้างมากดังภาพประกอบ 6 จึงเลือกใช้ Leave One Out Cross-validation ร่วมกับ Random forest regressor โดยการเลือกข้อมูลออกมาทีละ Offer เพื่อใช้เป็น Test set และใช้ข้อมูลที่เหลือเป็นชุดข้อมูลฝึกสอนโดยในชุดข้อมูลทดสอบจะมีข้อมูลทั้งหมด 24 Offer จะได้ทั้งหมด 24 Model โดยเลือกโมเดลที่ดีที่สุด 2 อันดับคือเลือกชุดข้อมูลที่เป็น Offer Id 1200581 และ 1199857 ออกมาแสดงดังตาราง 9

ตาราง 9 คะแนนสูงสุด 2 อันดับแรกจาก 24 Model

| Offer Id | Public Score | Private Score |
|---|---|---|
| 1200581 | 0.61570 | 0.60964 |
| 1199857 | 0.61556 | 0.60947 |

## 3.4 การปรับจูนพารามิเตอร์

ในงานวิจัยนี้ผู้วิจัยใช้ Grid Search Cross-Validation ในการหาพารามิเตอร์ที่เหมาะสมสำหรับแต่ละอัลกอริทึมโดยมีรายละเอียดดังตาราง 10 โดยช่วงของพารามิเตอร์ที่เลือกใช้นั้น ผู้วิจัยเลือกช่วงที่ครอบคลุมพารามิเตอร์ของหลายงานวิจัยและพารามิเตอร์จากงานวิจัยของ Nikulin [7] เพื่อหาพารามิเตอร์ที่เหมาะสมกับแบบจำลองการทำนายที่สร้างขึ้น

### 3.4.1 Random forest regressor

ในส่วนของการปรับจูนพารามิเตอร์เพื่อทำนายการกลับมาซื้อซ้ำของผู้ซื้อในเบื้องต้น กำหนดค่าพารามิเตอร์สำหรับ Random forest regressor ดังนี้

1. n_estimators: 100, 150, 200, 250, 300
2. max_depth: 10, 12, 14
3. min_samples_split: 6, 8, 10
4. min_samples_leaf: 3, 5, 7
5. random_states: 42

สาเหตุที่เลือกใช้ random_states 42 เนื่องจากต้องการให้ได้ผลลัพธ์เหมือนเดิมตลอด เมื่อลองปรับเปลี่ยนค่าอื่น ๆ และเลือกใช้ช่วง n_estimators, max_depth, min_samples_split, min_samples_leaf เพราะหากเลือกใช้ค่าที่มากกว่านี้จะทำให้เกิด Overfitting ทำให้ประสิทธิภาพของแบบจำลองการทำนายลดลง

**3.4.2 Random forest classifier**

ในส่วนของการปรับจูนพารามิเตอร์เพื่อทำนายการกลับมาซื้อซ้ำของผู้ซื้อในเบื้องต้น กำหนดค่าพารามิเตอร์สำหรับ Random forest classifier ดังนี้

1. n_estimators: 100, 150, 200, 250, 300
2. max_depth: 10, 12, 14
3. min_samples_split: 6, 8, 10
4. min_samples_leaf: 3, 5, 7

สาเหตุที่เลือกใช้ช่วง n_estimators, max_depth, min_samples_split, min_samples_leaf เพราะหากเลือกใช้ค่าที่มากกว่านี้จะทำให้เกิด Overfitting ทำให้ประสิทธิภาพของแบบจำลองการทำนายลดลง

**3.4.3 XGBoost**

ในส่วนของการปรับจูนพารามิเตอร์เพื่อทำนายการกลับมาซื้อซ้ำของผู้ซื้อในเบื้องต้นจะตั้งค่าพารามิเตอร์สำหรับ XGBoost ดังนี้

1. learning_rate: 0.08, 0.09, 0.10
2. max_depth: 6, 8, 10, 12
3. n_estimators: 150, 200, 250, 300, 400

สาเหตุที่เลือกใช้พารามิเตอร์ n_estimators, max_depth และ learning_rate ในช่วงนี้เนื่องจากหากเลือกค่าที่สูงกว่านี้จะทำให้เกิด Overfitting ทำให้ประสิทธิภาพแบบจำลองการทำนายลดลง

**3.4.4 Gradient Boost Regressor**

ในส่วนของการปรับจูนพารามิเตอร์เพื่อทำนายการกลับมาซื้อซ้ำของผู้ซื้อในเบื้องต้นจะตั้งค่าพารามิเตอร์สำหรับ XGBoost ดังนี้

1. learning_rate: [0.08, 0.09, 0.10]
2. max_depth: [5, 6, 7]
3. n_estimators: [200, 300, 400, 500]

สาเหตุที่เลือกใช้พารามิเตอร์ n_estimators, max_depth และ learning_rate ในช่วงนี้เนื่องจากหากเลือกค่าที่สูงกว่านี้จะทำให้เกิด Overfitting ทำให้ประสิทธิภาพแบบจำลองการทำนายลดลง

ตาราง 10 hyperparameter ของแต่ละอัลกอริทึม

| Algorithm | Parameters |
|---|---|
| Random forest regressor | n_estimator: [100, 150, 200, 250, 300]<br>max_depth: [10, 12, 14]<br>min_samples_split: [6, 8, 10]<br>min_samples_leaf: [3, 5, 7]<br>random_states: [42] |
| Random forest classifier | n_estimator: [100, 150, 200, 250, 300]<br>max_depth: [10, 12, 14]<br>min_samples_split: [6, 8, 10]<br>min_samples_leaf: [3, 5, 7] |
| XGBoost | learning_rate: [0.08, 0.09, 0.10]<br>max_depth: [6, 8, 10, 12]<br>n_estimators: [150, 200, 250, 300, 400] |
| Gradient Boost Regressor | learning_rate: [0.08, 0.09, 0.10]<br>max_depth: [5, 6, 7]<br>n_estimators: [200, 300, 400, 500] |

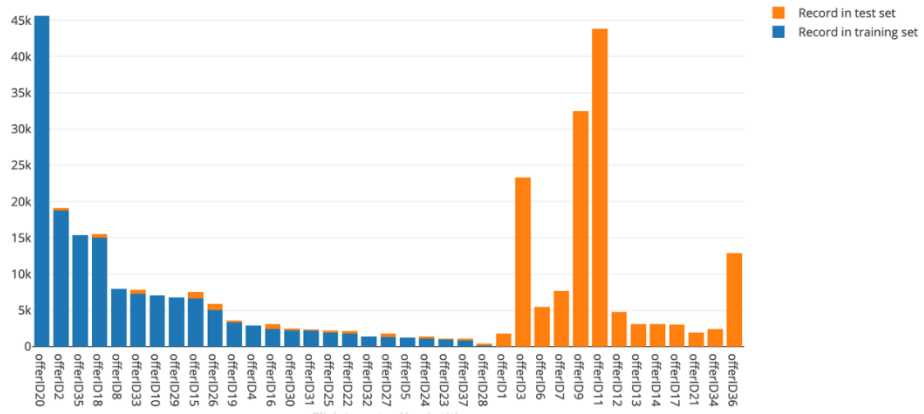
# บทที่ 4
# ผลลัพธ์จากการวิจัย

ในการวิจัยเรื่องการทำนายการซื้อซ้ำของผู้ซื้อโดยเทคนิคการเรียนรู้ของเครื่อง ผู้วิจัยได้ดำเนินการวิจัยโดยศึกษาตามขั้นตอนต่าง ๆ ตลอดจนการวัดประสิทธิภาพเพื่อให้บรรลุวัตถุประสงค์ของการวิจัยที่ได้กำหนดไว้ ได้ดังนี้

1. ผลลัพธ์ของการวิเคราะห์ข้อมูล
2. ผลลัพธ์ของการวิเคราะห์ฟีเจอร์
3. ผลลัพธ์การสร้างแบบจำลองการทำนายโดยใช้ข้อมูลประวัติการซื้อสินค้า
4. ผลลัพธ์ของการเปรียบเทียบฟีเจอร์
5. ผลลัพธ์ของแบบจำลองที่ใช้เทคนิค Leave One Offer Out

## ผลลัพธ์ของการวิเคราะห์ข้อมูล

จากการสำรวจพบว่าข้อมูลผู้ซื้อและข้อมูลโปรโมชั่นในชุดข้อมูลฝึกสอน (training set) และชุดข้อมูลทดสอบ (test set) มีความแตกต่างกันมากแสดงดังภาพประกอบ 6 โดยแกน Y คือจำนวนข้อมูลประวัติโปรโมชั่นที่ผู้ซื้อเคยได้รับของแต่ละโปรโมชั่นและแกน X คือโปรโมชั่น (Offer ID) ผู้วิจัยจึงใช้เทคนิค Leave One Offer Out มาช่วยในการสร้างแบบจำลองการทำนายเผื่อให้เกิดสถานการณ์ที่เหมือนจริง




ภาพประกอบ 6 ความแตกต่างของข้อมูลโปรโมชั่นในชุดข้อมูลฝึกสอนและชุดข้อมูลทดสอบ

## ผลลัพธ์การวิเคราะห์ฟีเจอร์

ฟีเจอร์ที่มีในชุดข้อมูล Kaggle “Acquire Valued Shopper Challenge” ผู้วิจัยได้นำมาเปรียบเทียบหาความสำคัญของฟีเจอร์โดยใช้ Random forest ของ scikit-learn ได้ผลลัพธ์ดังตาราง 11 และเมื่อพิจารณาค่าความสำคัญของแต่ละฟีเจอร์พบว่าฟีเจอร์ที่มีความสำคัญหรือผลกระทบต่อผลลัพธ์การทำนายมากที่สุดได้แก่ Company, Category และ Brand ซึ่งผลลัพธ์ที่ได้เหมือนกับงานของ Nikulin [7] จากผลลัพธ์ที่ได้จึงสร้างฟีเจอร์จากการรวมกันของ 3 ฟีเจอร์ข้างต้นเรียกว่า “Product” และสร้างฟีเจอร์จากหลักการ Content-Based และ Collaborative Filtering

ตาราง 11 ความสำคัญของฟีเจอร์ (Feature Importance)

| N | Feature | Importance |
|---|---|---|
| 1 | Company | 0.311937 |
| 2 | Category | 0.196459 |
| 3 | Brand | 0.172855 |
| 4 | Chain | 0.118947 |
| 5 | Offer Value | 0.103365 |
| 6 | Market | 0.093643 |

## ผลลัพธ์ของการสร้างแบบจำลองโดยใช้ Base features และ All features

ในการสร้างฟีเจอร์ผู้วิจัยได้หาค่า Feature Importance ของแต่ละฟีเจอร์ตามตาราง 11 เพื่อคัดเลือกฟีเจอร์มีที่ความสำคัญมาสร้างเป็นฟีเจอร์ใหม่ เพื่อใช้ในการสร้างแบบจำลองในแต่ละอัลกอริทึม ได้แก่ Random forest regressor, Random forest classifier, eXtreme Gradient Boost (XGBoost) และ Gradient Boost โดยใช้ฟีเจอร์ที่สร้างขึ้นทั้ง 2 กลุ่ม โดยพบว่าเมื่อใช้ฟีเจอร์ All features พบว่าผลลัพธ์ของแบบจำลองจะให้ค่า Cross-validation Score (CV), Public AUC Score (PB) และ Private AUC Score (PV) ที่สูงกว่าการใช้เพียงฟีเจอร์ Base features เพียงอย่างเดียวตามตาราง 12

ตาราง 12 ผลลัพธ์การทำนายโดยใช้ Base features และ All features ของแต่ละอัลกอริทึม

| Used | Random forest regressor | | | Random forest classifier | | | XGBoost | | | Gradient Boost | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Features | CV | PB | PV | CV | PB | PV | CV | PB | PV | CV | PB | PV |
| Base features | 0.56580 | 0.60053 | 0.59511 | 0.64967 | 0.54256 | 0.53740 | 0.70724 | 0.58055 | 0.57749 | 0.70600 | 0.58612 | 0.58176 |
| All features | 0.58109 | 0.61119 | 0.60669 | 0.65971 | 0.55521 | 0.56043 | 0.71365 | 0.56878 | 0.56721 | 0.71410 | 0.60222 | 0.59898 |

## ผลลัพธ์ของการปรับจูนพารามิเตอร์ของแบบจำลอง

จากผลลัพธ์ก่อนหน้าผู้วิจัยได้ปรับจูนพารามิเตอร์ของแบบจำลองการทำนายแต่ละอัลกอริทึมโดยการใช้ Grid Search Cross-validation ในการหาพารามิเตอร์ที่เหมาะสมและทดสอบกับพารามิเตอร์ของ Nikulin [7] เพื่อเพิ่มประสิทธิภาพแบบจำลองการทำนายโดยมีรายละเอียดพารามิเตอร์ดังตาราง 13 และมีผลลัพธ์แสดงดังตาราง 14

## ผลลัพธ์ของแบบจำลองเมื่อใช้เทคนิค Leave One Offer Out

ผู้วิจัยได้ใช้เทคนิค Leave One Offer Out ร่วมกับฟีเจอร์ทั้งหมดที่สร้างขึ้น (All Features) ในการสร้างแบบจำลองการทำนายแต่ละอัลกอริทึมโดยมีผลลัพธ์การทำ Cross-Validation (CV) โดยใช้ K-fold 5 และนำผลลัพธ์จากแบบจำลองที่ได้ไปทดสอบเพื่อทำการวัดผลประสิทธิภาพใน Kaggle ด้วยค่า AUC Score ได้แก่ Public Score (PB), Private Score (PV) โดยผลลัพธ์ของแต่ละอัลกอริทึมแสดงดังตาราง 15

ตาราง 13 พารามิเตอร์ที่ใช้หลังการปรับจูน

| **Algorithm** | **Parameters** |
|---|---|
| Random Forest Regressor | n_estimator = 300, criterion='mse', max_depth=12, min_samples_split=10, max_features ='sqrt', min_samples_leaf=3 |
| Random Forest Classifier | n_estimator = 200, max_depth=12, min_samples_split=10, min_samples_leaf=3 |
| XGBoost | learning_rate: 0.08, max_depth: 10, n_estimators: 400 |
| Gradient Boost Regressor | learning_rate: 0.08, max_depth: 5, n_estimators: 500 |

ตาราง 14 ผลลัพธ์ของแบบจำลองการทำนายแต่ละอัลกอริทึมหลังปรับจูนพารามิเตอร์

| Used Features | Random forest regressor | | | Random forest classifier | | | XGBoost | | | Gradient Boost | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CV | PB | PV | CV | PB | PV | CV | PB | PV | CV | PB | PV |
| Base features | 0.70420 | 0.60053 | 0.59511 | 0.70444 | 0.59717 | 0.59359 | 0.70432 | 0.54839 | 0.55297 | 0.70750 | 0.57266 | 0.57106 |
| All features | 0.71459 | 0.61515 | 0.60759 | 0.64909 | 0.61731 | 0.60956 | 0.70934 | 0.57477 | 0.56837 | 0.71437 | 0.60207 | 0.59851 |

ตาราง 15 ผลลัพธ์ของแบบจำลองการทำนายแต่ละอัลกอริทึมเมื่อใช้เทคนิค Leave One Offer Out

| Used Features | Random forest regressor | | | Random forest classifier | | | XGBoost | | | Gradient Boost | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CV | PB | PV | CV | PB | PV | CV | PB | PV | CV | PB | PV |
| All features + Leave One Offer Out | 0.75020 | 0.61574 | 0.60880 | 0.60730 | 0.61296 | 0.60731 | 0.70670 | 0.60623 | 0.60311 | 0.71200 | 0.58755 | 0.57638 |

# บทที่ 5
# สรุปผลการวิจัย อภิปรายผล และข้อเสนอแนะ

งานวิจัยนี้นำเสนอการสร้างแบบจำลองการทำนายการซื้อซ้ำของผู้ซื้อโดยใช้เทคนิคการเรียนรู้ของเครื่อง โดยนิยามปัญหาคือต้องการทำนายผู้ซื้อคนใดจะเป็นผู้ซื้อที่มีความภักดีต่อยี่ห้อสินค้า (loyal customer) ที่มีโอกาสจะกลับมาซื้อสินค้า เมื่อมีการเสนอโปรโมชั่นหนึ่ง ๆ หรือจะเป็นลูกค้าที่มาซื้อเพียงครั้งเดียวเพื่อใช้โปรโมชั่น (one-time deal hunter) ต้องการจำแนกผู้ซื้อที่จะกลับมาซื้อสินค้าซ้ำในอนาคตหลังจากได้รับโปรโมชั่นไปแล้ว โดยผู้วิจัยได้วัดประสิทธิภาพของแบบจำลองแต่ละอัลกอริทึมและการทำ Feature Engineering รูปแบบต่าง ๆ เพื่อนำผลลัพธ์มาเปรียบเทียบและสรุปผลโดยสามารถแบ่งหัวข้อในการสรุปได้ดังต่อไปนี้

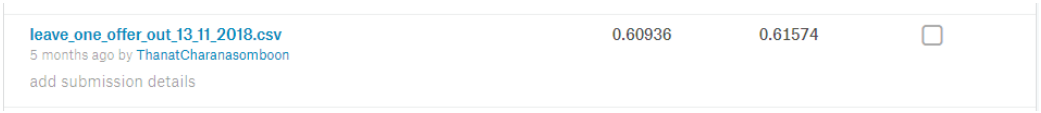
1. สรุปผลการวิจัย
2. อภิปรายผลการวิจัย
3. ข้อเสนอแนะ

## 5.1 สรุปผลการวิจัย

การทำนายการซื้อซ้ำของผู้ซื้อจะช่วยให้ธุรกิจร้านค้าสามารถลดต้นทุนและช่วยในการจัดทำแผนการตลาดในอนาคต

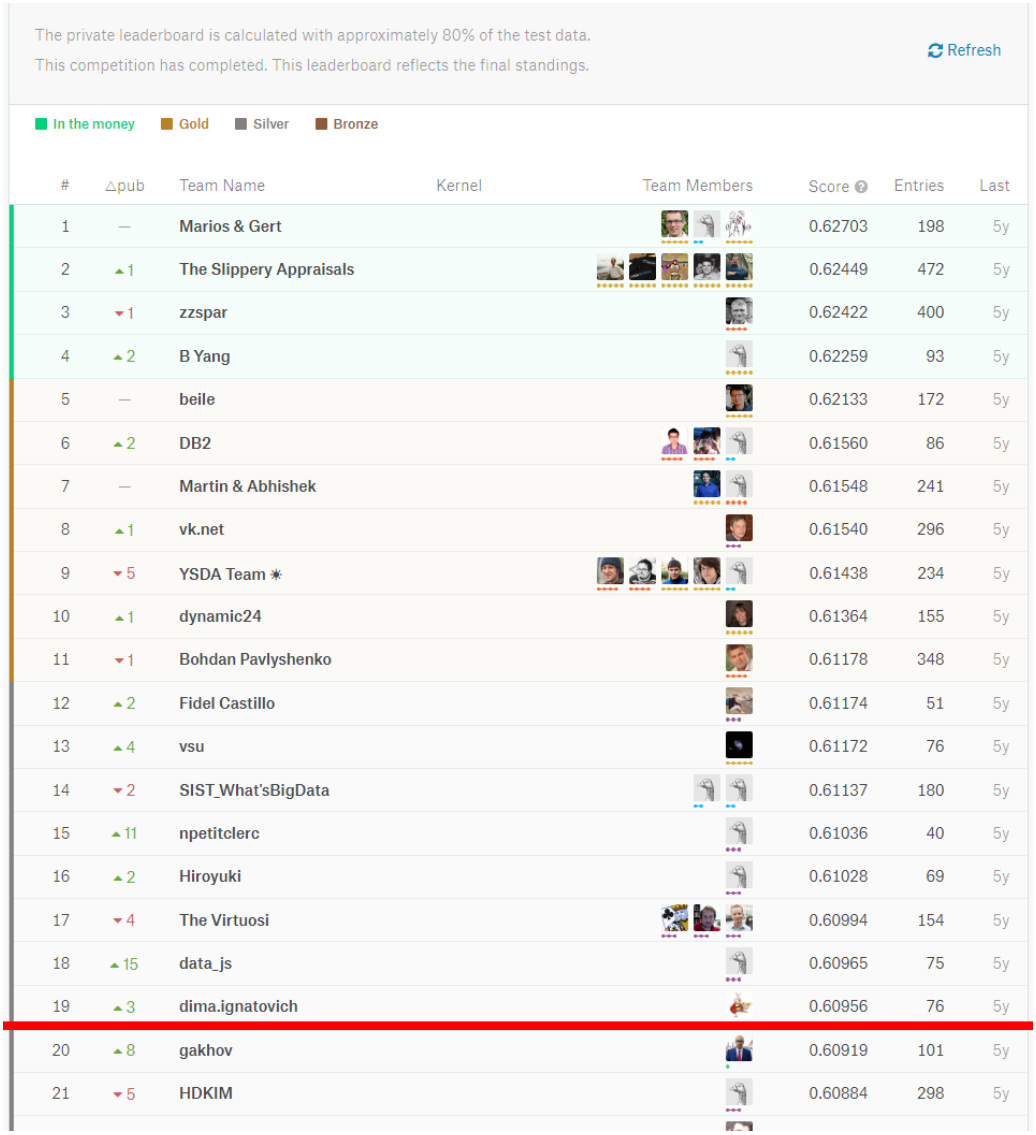
ในงานวิจัยนี้ผู้วิจัยได้ใช้ชุดข้อมูล “Acquire Valued Shopper Challenge” จากการแข่งขันในเว็บไซต์ Kaggle โดยผู้วิจัยได้สร้างฟีเจอร์จากการรวบรวมข้อมูล (Aggregate) ในช่วงเวลาที่สนใจเพื่อลดขนาดพื้นที่ในการจัดเก็บข้อมูล โดยแบ่งเป็น 2 กลุ่ม ได้แก่ (1) Base features และ (2) All features โดยสร้างประวัติการซื้อสินค้าในของลูกค้าโดยให้ความสำคัญกับแบรนด์ (brand) หมวดหมู่ (category) และบริษัท (company) โดยใช้อัลกอริทึมที่หลากหลาย ได้แก่ random forest regressor, random forest classifier, XGBoost, Gradient Boost นอกจากนี้ผู้วิจัยได้ใช้เทคนิค Leave One Offer Out เพื่อให้ได้ผลลัพธ์การทำนายที่ดีขึ้น

จากการทดลองพบว่า Random forest regressor ร่วมกับ Leave One Offer Out ให้ผลลัพธ์ที่ดีกว่าวิธีอื่น เมื่อนำผลลัพธ์ที่ได้ Submit ที่ Kaggle ได้ Public AUC Score 0.61574 และ Private AUC Score 0.60936 ตามภาพประกอบ 7 หรือเทียบเท่ากับอันดับที่ 20 บน Private Leader Board แสดงดังภาพประกอบ 8




ภาพประกอบ 7 ผลลัพธ์ที่ได้จากการส่งผลในการแข่งขัน Kaggle “Acquire Valued Shoppers”

ที่มา https://www.kaggle.com/c/acquire-valued-shoppers-challenge/leaderboard




ภาพประกอบ 8 Private Leaderboard

ที่มา https://www.kaggle.com/c/acquire-valued-shoppers-challenge/leaderboard

**5.2 อภิปรายผลการวิจัย**

จากการตั้งสมมติฐานหากใช้ All Features ในการสร้างแบบจำลองการทำนายจะให้ผลลัพธ์ที่ดีขึ้นเมื่อเทียบกับการใช้ Base Features อย่างเดียวและพบว่า Random Forest Regressor ใช้ร่วมกับ Leave One Offer Out ให้ผลลัพธ์สูงที่สุดเมื่อเทียบกับอัลกอริทึมอื่นโดยวิธีการนี้สามารถใช้เป็นแนวทางในการสร้างแบบจำลองการทำนายซื้อซ้ำในอนาคตได้

**5.3 ข้อเสนอแนะ**

เนื่องจากข้อมูล Transaction มีปริมาณมากทำให้การเตรียมข้อมูล (Data preparation) เพื่อจะสร้างแบบจำลองการทำนายต้องใช้เวลานาน ควรมีการเก็บข้อมูลในรูปแบบที่พร้อมใช้งานมีการรวมข้อมูล (Aggregation) ในรูปแบบที่เหมาะสมในการใช้สร้างโมเดลเลยเพื่อช่วยลดพื้นที่ที่ต้องใช้ในการเก็บข้อมูลและลดเวลาในการสร้างแบบจำลองการทำนาย

# ภาคผนวก

งานวิจัยนี้พัฒนาแบบจำลองการทำนายโดยใช้ภาษา Python เวอร์ชั่น 2.7 ทั้งนี้ผู้วิจัยได้แสดงโค้ดตัวอย่างสำหรับการสร้างฟีเจอร์ทั้งหมดใน โดยโค้ดส่วนของการสร้างฟีเจอร์ทั้งหมดสามารถดูได้จาก https://bit.ly/2KSGUwy และตัวอย่างโค้ดบางส่วนที่สำคัญเพื่อใช้ประกอบงานวิจัยโดยมีรายละเอียดดังนี้

1. โค้ดใช้สำหรับลดขนาดข้อมูลโดยการเลือกข้อมูลที่มีความเกี่ยวข้องกับโปรโมชั่นที่เราสนใจดังภาพประกอบ 9

```
def reduce_data(offers_file, transactions_file, reduced_file):
        start = datetime.now()
        #get all categories and comps on offer in a dict
        offers_cat = {}
        offers_co = {}
        for e, line in enumerate( open(offers_file) ):
                offers_cat[ line.split(",")[1] ] = 1
                offers_co[ line.split(",")[3] ] = 1
        #open output file
        with open(reduced_file, "wb") as outfile:
                #go through transactions file and reduce
                reduced = 0
                for e, line in enumerate( open(transactions_file) ):
                        if e == 0:
                                outfile.write( line ) #print header
                        else:
                                #only write when if category in offers dict
                                if line.split(",")[3] in offers_cat or line.split(",")[4] in offers_co:
                                        outfile.write( line )
                                        reduced += 1
                        #progress
                        if e % 5000000 == 0:
                                print e, reduced, datetime.now() - start
        print e, reduced, datetime.now() - start
```

ภาพประกอบ 9 โค้ดใช้สำหรับลดขนาดข้อมูล

2. โค้ดการหาความสำคัญของฟีเจอร์ (Feature importance) แสดงดังภาพประกอบ 10

```
clf = RandomForestRegressor(n_estimators=200,criterion='mse', max_depth=12,
                            min_samples_split=10,max_features ='sqrt', min_samples_leaf=3)
clf.fit(X_train, y_train)
pred = clf.predict(X_test)

clf.feature_importances_
```

ภาพประกอบ 10 โค้ดการหาความสำคัญของฟีเจอร์ (Feature importance)

3. โค้ดสำหรับการสร้างกราฟแสดงจำนวนข้อมูลในชุดข้อมูลฝึกสอนและข้อมูลทดสอบในแต่ละโปรโมชั่นแสดงดังภาพประกอบ 11

```
import plotly.plotly as py
import plotly.graph_objs as go

trace1 = go.Bar(
    x=xtrain,
    y=ytrain,
    name='Record in training set'
)
trace2 = go.Bar(
    x=xtest,
    y=ytest,
    name='Record in test set'
)

data = [trace1, trace2]

layout = go.Layout(
    barmode='stack'
)

fig = go.Figure(data=data, layout=layout)
py.iplot(fig, filename='stacked-bar')
```

ภาพประกอบ 11 โค้ดสำหรับการสร้างกราฟแสดงจำนวนข้อมูลในชุดข้อมูลฝึกสอนและข้อมูลทดสอบในแต่ละโปรโมชั่น

4. โค้ดสำหรับสร้าง Base features แสดงดังภาพประกอบ 12 ภาพประกอบ 13 และภาพประกอบ 14

```
def generate_features(transactions_file, out_file):
        #keep a dictionary with the offer data
        offers = {}
        offers_categories = {}
        offers_companies = {}
        for e, line in enumerate( open(offers_file) ):
                row = line.strip().split(",")
                offers[ row[0] ] = row
                offers_categories[row[1]] = 1
                offers_companies[row[3]] = 1

        # dicts with variables from history
        ids = {}
        for e, line in enumerate( open(history_file) ):
                if e > 0:
                        row = line.strip().split(",")
                        ids[row[0]] = row

        seen_ids = set([])

        outfile = open(out_file, "wb")
        outfile.write("label repeattrips id "+string.join(feature_list)+" market chain\n")

        #iterate through reduced dataset
        last_id = 0
        features = defaultdict(float)
        for e, line in enumerate( open(transactions_file) ):
                if e > 0: #skip header
                        #poor man's csv reader
                        row = line.strip().split(",")
                        #write away the features when we get to a new shopper id
                        if last_id != row[0] and e != 1:

                                #generate negative features
                                if "has_bought_company" not in features:
                                        features['never_bought_company'] = 1

                                if "has_bought_category" not in features:
                                        features['never_bought_category'] = 1

                                if "has_bought_brand" not in features:
                                        features['never_bought_brand'] = 1

                                if "has_bought_brand" in features and "has_bought_category" in features and "has_bought_company"
in features:
                                        features['has_bought_brand_company_category'] = 1

                                if "has_bought_brand" in features and "has_bought_category" in features:
                                        features['has_bought_brand_category'] = 1

                                if "has_bought_brand" in features and "has_bought_company" in features:
                                        features['has_bought_brand_company'] = 1

                                outline = ""
                                if not testset and last_id in ids:
                                        outline += str(features["label"]) + " " + ids[last_id][4] + " " + str(last_id)
                                else:
                                        outline += "-1 -1 "+str(last_id)
                                for l in feature_list:
                                        if l in features:
                                                outline += " "+str(features[l])
                                        else:
                                                outline += " 0"
                                # write chain and market
                                if last_id in ids:
                                        outline += " "+ids[last_id][3]
                                        outline += " "+ids[last_id][1]
                                outline += "\n"
                                if last_id in ids:
                                        outfile.write( outline )
                                        seen_ids.add(last_id)
                                #reset features
                                features = defaultdict(float)
                        #check if we have a valid sample
                        if row[0] in ids:
                                #generate label and history
                                history = ids[row[0]]
                                if not testset and row[0] in ids:
                                        if ids[row[0]][5] == "t":
                                                features['label'] = 1
                                        else:
                                                features['label'] = 0
```

ภาพประกอบ 12 โค้ดสำหรับสร้าง Base features

```
                        features['offer_value'] = offers[ history[2] ][4]
                        features['offer_id'] = history[2]

                        offervalue = offers[ history[2] ][4]

                        features['total_spend_all'] += float( row[10] )

                        if row[3] in offers_categories or row[4] in offers_companies:
                                features['total_spend_ccb'] += float( row[10] )

                        if offers[ history[2] ][3] == row[4]:
                                features['has_bought_company'] += 1.0
                                features['has_bought_company_q'] += float( row[9] )
                                features['has_bought_company_a'] += float( row[10] )

                                date_diff_days = diff_days(row[6],history[-1])
                                if date_diff_days < 30:
                                        features['has_bought_company_30'] += 1.0
                                        features['has_bought_company_q_30'] += float( row[9] )
                                        features['has_bought_company_a_30'] += float( row[10] )
                                if date_diff_days < 60:
                                        features['has_bought_company_60'] += 1.0
                                        features['has_bought_company_q_60'] += float( row[9] )
                                        features['has_bought_company_a_60'] += float( row[10] )
                                if date_diff_days < 90:
                                        features['has_bought_company_90'] += 1.0
                                        features['has_bought_company_q_90'] += float( row[9] )
                                        features['has_bought_company_a_90'] += float( row[10] )
                                if date_diff_days < 180:
                                        features['has_bought_company_180'] += 1.0
                                        features['has_bought_company_q_180'] += float( row[9] )
                                        features['has_bought_company_a_180'] += float( row[10] )

                        if offers[ history[2] ][1] == row[3]:

                                features['has_bought_category'] += 1.0
                                features['has_bought_category_q'] += float( row[9] )
                                features['has_bought_category_a'] += float( row[10] )
                                date_diff_days = diff_days(row[6],history[-1])
                                if date_diff_days < 30:
                                        features['has_bought_category_30'] += 1.0
                                        features['has_bought_category_q_30'] += float( row[9] )
                                        features['has_bought_category_a_30'] += float( row[10] )
                                if date_diff_days < 60:
                                        features['has_bought_category_60'] += 1.0
                                        features['has_bought_category_q_60'] += float( row[9] )
                                        features['has_bought_category_a_60'] += float( row[10] )
                                if date_diff_days < 90:
                                        features['has_bought_category_90'] += 1.0
                                        features['has_bought_category_q_90'] += float( row[9] )
                                        features['has_bought_category_a_90'] += float( row[10] )
                                if date_diff_days < 180:
                                        features['has_bought_category_180'] += 1.0
                                        features['has_bought_category_q_180'] += float( row[9] )
                                        features['has_bought_category_a_180'] += float( row[10] )
                        if offers[ history[2] ][5] == row[5] and (row[3] in offers_categories or row[4] in offers_compan
ies):
                                features['has_bought_brand'] += 1.0
                                features['has_bought_brand_q'] += float( row[9] )
                                features['has_bought_brand_a'] += float( row[10] )
                                date_diff_days = diff_days(row[6],history[-1])
                                if date_diff_days < 30:
                                        features['has_bought_brand_30'] += 1.0
                                        features['has_bought_brand_q_30'] += float( row[9] )
                                        features['has_bought_brand_a_30'] += float( row[10] )
                                if date_diff_days < 60:
                                        features['has_bought_brand_60'] += 1.0
                                        features['has_bought_brand_q_60'] += float( row[9] )
                                        features['has_bought_brand_a_60'] += float( row[10] )
                                if date_diff_days < 90:
                                        features['has_bought_brand_90'] += 1.0
                                        features['has_bought_brand_q_90'] += float( row[9] )
                                        features['has_bought_brand_a_90'] += float( row[10] )
                                if date_diff_days < 180:
                                        features['has_bought_brand_180'] += 1.0
                                        features['has_bought_brand_q_180'] += float( row[9] )
                                        features['has_bought_brand_a_180'] += float( row[10] )
                last_id = row[0]
                if e % 100000 == 0:
                        print e
        # do stuff for ids without transactions
        allids = set(ids.keys())
        unseen_ids = allids.difference(seen_ids)
        for ui in unseen_ids:
                features = defaultdict(float)
                history = ids[ui]
                features['offer_value'] = offers[ history[2] ][4]
                features['offer_id'] = history[2]
                if not testset:
                        if ids[ui][5] == "t":
                                features['label'] = 1
                        else:
                                features['label'] = 0
```

ภาพประกอบ 13 โค้ดสำหรับสร้าง Base features (ต่อ)

```
                if "has_bought_company" not in features:
                        features['never_bought_company'] = 1
                if "has_bought_category" not in features:
                        features['never_bought_category'] = 1
                if "has_bought_brand" not in features:
                        features['never_bought_brand'] = 1
                if "has_bought_brand" in features and "has_bought_category" in features and "has_bought_company" in features:
                        features['has_bought_brand_company_category'] = 1
                if "has_bought_brand" in features and "has_bought_category" in features:
                        features['has_bought_brand_category'] = 1
                if "has_bought_brand" in features and "has_bought_company" in features:
                        features['has_bought_brand_company'] = 1
                outline = ""
                if not testset:
                        outline += str(features["label"]) + " " + ids[ui][4] + " " + str(ui)
                else:
                        outline += "-1 -1 "+str(ui)
                for l in feature_list:
                        if l in features:
                                outline += " "+str(features[l])
                        else:
                                outline += " 0"
                # write chain and market
                outline += " "+ids[ui][3]
                outline += " "+ids[ui][1]
                outline += "\n"
                outfile.write( outline )

if __name__ == '__main__':
        reduce_data(offers_file, transactions_file, reduced_file)
        generate_features(transactions_file, out_file)
```

ภาพประกอบ 14 โค้ดสำหรับสร้าง Base features (ต่อ 2)

5. โค้ดสำหรับนับจำนวนวันตั้งแต่การซื้อครั้งแรกแสดงดังภาพประกอบ 15

```
# create features on time since first transaction
import string,datetime, os

testset = False
if testset:
        folder = "./test/"
        history = "../data/testHistory.csv"
else:
        folder = "./train/"
        history = "../data/trainHistory.csv"

user_dates = {}
fi = open("../data/user_dates.csv","r")
fi.next()
for lines in fi:
        li = lines.strip().split(",")
        startdate = datetime.datetime.strptime(li[1],"%Y-%m-%d").date()
        #enddate = datetime.datetime.strptime(li[2],"%Y-%m-%d").date()
        user_dates[li[0]] = startdate

fi = open(history, "r")
fi.next()
of = open(os.path.join(folder, "first_transaction_features.csv"),"w")
of.write("id,days_since_first_transaction\n")
for lines in fi:
        li = lines.strip().split(",")
        uid = li[0]
        if testset:
                offerdate = datetime.datetime.strptime(li[4],"%Y-%m-%d").date()
        else:
                offerdate = datetime.datetime.strptime(li[6],"%Y-%m-%d").date()
        days_since_first = (offerdate-user_dates[uid]).days
        of.write(uid+","+str(days_since_first)+"\n")
```

ภาพประกอบ 15 โค้ดสำหรับสร้างฟีเจอร์นับวันตั้งแต่การซื้อครั้งแรก

6. โค้ดสำหรับสร้างฟีเจอร์ในทางลบ (negative features) แสดงดังภาพประกอบ 16 และภาพประกอบ 17

```
# get "negative" features:
# bought product just once
# returned product

import datetime,string,os
import numpy as np
from collections import defaultdict

testset = False
if testset:
        folder = "./test/"
else:
        folder = "./train/"

# get products belonging to offer
offers = {}
fi = open("../data/offers.csv","r")
fi.next()
for lines in fi:
        li = lines.strip().split(",")
        offers[li[0]] = li[1]+" "+li[3]+" "+li[5]
        # li[0] = offer_id, li[1] = category, li[2] = quantity, li[3] = company, li[5] = brand

# get history information
history = {}
fi = open("../data/trainHistory.csv","r")
fi.next()
for lines in fi:
        li = lines.strip().split(",")
        history[li[0]] = li # li[0] = id
fi = open("../data/testHistory.csv","r")
fi.next()
for lines in fi:
        li = lines.strip().split(",")
        history[li[0]] = li # li[0] = id
# offer is li[2] here

user_dates = {}
fi = open("../data/user_dates.csv","r")
fi.next()
for lines in fi:
        li = lines.strip().split(",")
        user_dates[li[0]] = li[2]

of = open( os.path.join(folder, "negative_features.csv"),"w")
of.write("id,returned_product,days_from_lastdata_until_offerdate\n")

fi = open("../data/transactions.csv","r")
fi.next()
# set standard variables
returned = False
lastid = 0

for e,lines in enumerate(fi):
        li = lines.strip().split(",")
        if not lastid == li[0] and e > 0:
                of.write(lastid+",")
                if returned:
                        of.write("1,")
                else:
                        of.write("0,")

                # get lastdate and offerdate
                offerstuff = history[lastid]
                if len(offerstuff) == 7:
                        offerdate = datetime.datetime.strptime(offerstuff[6],"%Y-%m-%d").date()
                else:
                        offerdate = datetime.datetime.strptime(offerstuff[4],"%Y-%m-%d").date()
                enddate = datetime.datetime.strptime(user_dates[lastid],"%Y-%m-%d").date()
                daydiff = (offerdate-enddate).days
```

ภาพประกอบ 16 โค้ดสร้างฟีเจอร์ในเชิงลบ (negative features)

```
        user = li[0]
        product = string.join(li[3:6]," ")
        pi = offers[history[li[0]][2]]

        if product == pi:
                if float(li[10]) < 0:
                        returned = True
        lastid = li[0]
        if e % 1000000 == 0 and e > 0:
                print e

# last entry
of.write(lastid+",")
if returned:
        of.write("1,")
else:
        of.write("0,")
# get lastdate and offerdate
offerstuff = history[lastid]
if len(offerstuff) == 7:
        offerdate = datetime.datetime.strptime(offerstuff[6],"%Y-%m-%d").date()
else:
        offerdate = datetime.datetime.strptime(offerstuff[4],"%Y-%m-%d").date()
enddate = datetime.datetime.strptime(user_dates[lastid],"%Y-%m-%d").date()
daydiff = (offerdate-enddate).days
of.write(str(daydiff)+"\n")
```

ภาพประกอบ 17 โค้ดสร้างฟีเจอร์ในเชิงลบ (negative features) (ต่อ)

7. โค้ดสำหรับสร้างฟีเจอร์โอกาสการกลับมาซื้อสินค้าแต่ละชิ้นแสดงดังภาพประกอบ 18

```
# get start and enddate of all users
user_dates = {}
fi = open("../data/user_dates.csv","r")
fi.next()
for lines in fi:
        li = lines.strip().split(",")
        user_dates[li[0]] = {'firstdate' : datetime.datetime.strptime(li[1],"%Y-%m-%d").date(),'lastdate' : datetime.datetime.st
rptime(li[2],"%Y-%m-%d").date() }

# get products we are interested in
productids = set([])
for e, line in enumerate( open("../data/offers.csv","r") ):
        if e > 0:
                li = line.strip().split(",")
                productids.add(li[1]+" "+li[3]+" "+li[5])
productids = list(productids)

of = open( os.path.join(folder, "product_repeat_buy_probability.csv"),"w")
of.write("product_id,repeat_buy_prob_30d,repeat_buy_prob_60d,repeat_buy_prob_90d\n")


repeaters_30d = defaultdict(list)
repeaters_60d = defaultdict(list)
repeaters_90d = defaultdict(list)

fi = open("../data/reduced.csv","r")
fi.next()
# set standard variables
bought_pre_6m = defaultdict(bool)
firstbuy_date = {}
repeatbuy_dates = defaultdict(list)
lastid = 0

for e,lines in enumerate(fi):
        li = lines.strip().split(",")
        if not lastid == li[0] and e > 0:
                # check if it's valid
                for pi in productids:
                        if pi in firstbuy_date:
                                if (user_dates[user]['lastdate']-firstbuy_date[pi]).days > 30:
                                        repeat_within_30d = 0
                                        # check if it's repeated within thirty days
                                        for d in repeatbuy_dates[pi]:
                                                if (d-firstbuy_date[pi]).days < 30:
                                                        repeat_within_30d = 1
                                        repeaters_30d[pi].append(repeat_within_30d)
```

ภาพประกอบ 18 โค้ดสำหรับสร้างฟีเจอร์โอกาสการกลับมาซื้อสินค้าแต่ละชิ้น

8. โค้ดสำหรับสร้างฟีเจอร์เทรนด์การใช้จ่ายในแต่ละช่วงของแต่ละหมวดหมู่แสดงดังภาพประกอบ 19

```
# create seasonal feature
import numpy as np
import pandas as pd
import datetime, os

testset = False
if testset:
        folder = "./test/"
        history_file = "../data/testHistory.csv"
else:
        folder = "./train/"
        history_file = "../data/trainHistory.csv"

# load seasonal_cat.csv
seasonal = pd.io.parsers.read_csv("../data/seasonal_cat.csv", index_col=0)
# drop data with low amount of customers
seasonal = seasonal[8:388]

earliest = datetime.date(2012,3,10)
latest = datetime.date(2013,3,24)

categories = ['706', '799', '1703', '1726', '2119', '2202', '3203', '3504', '3509', '4401', '4517', '5122', '5824', '5558', '561
6', '5619', '6202', '7205', '9115', '9909']

# calculate trendlines for each category
cat_trend = {}
for c in categories:
  # divide spending in 2013-03-05 - 2013-03-19 by spending in 2012-03-05 - 2012-03-19
  avg2012 = seasonal[c][0:15].mean()
  avg2013 = seasonal[c][365:380].mean()
  cat_trend[c] = avg2013/avg2012

# remove estimated trends from spending (i.e. only get seasonal effects)
for c in categories:
        div = 1. + (cat_trend[c]-1.)*np.array(range(380))/365.
        seasonal[c] = seasonal[c]/div

# sum over similar dates
for c in categories:
        seasonal[c][0:15] = (seasonal[c][0:15]*seasonal['num_customers'][0:15]).values + (seasonal[c][365:380]*seasonal['num_cus
tomers'][365:380]).values
        seasonal[c][0:15] /= ( (seasonal['num_customers'][0:15]).values+(seasonal['num_customers'][365:380]).values )
# remove extraneous dates
seasonal = seasonal[0:365]

# get 30 day spending average for each category
cat_spend_avg = {}
for c in categories:
        cat_spend_avg[c] = seasonal[c].mean()*30.

# get offers (for category)
offer_data = {}
fi = open("../data/offers.csv","r")
fi.next()
for lines in fi:
        li = lines.strip().split(",")
        offer_data[li[0]] = li[1]

fi = open( history_file,"r")
fi.next()
of = open( os.path.join(folder, "seasonal_features.csv"),"w")
of.write("id,seasonal_spend_rate_30d,seasonal_spend_rate_30d_no_trend\n")
for lines in fi:
        li = lines.strip().split(",")
        if testset:
                offerdate = datetime.datetime.strptime(li[4],"%Y-%m-%d").date()
        else:
                offerdate = datetime.datetime.strptime(li[6],"%Y-%m-%d").date()
        offerdateindex = (offerdate-earliest).days
        avg30d = 0.0
        category = offer_data[li[2]]
        # average spending month after offerdate for the category
        for r in range(30):
                avg30d += seasonal[category][(offerdateindex+r) % 365]
        avg30d /= cat_spend_avg[category]
        # then multiply this by the extrapolated trendline from the year ( i.e. (days since 2012-03-12 * trend/365) )
        avg30d_w_trend = avg30d * (offerdateindex+r)/365.*cat_trend[category]
        of.write(li[0]+","+str(avg30d_w_trend)+","+str(avg30d)+"\n")
```

ภาพประกอบ 19 โค้ดสำหรับสร้างฟีเจอร์เทรนด์การใช้จ่ายในแต่ละช่วงของแต่ละหมวดหมู่

9. โค้ดสำหรับสร้างฟีเจอร์เปรียบเทียบราคาสินค้ากับสินค้าที่คล้ายกันแสดงดังภาพประกอบ 20 และภาพประกอบ 21

```
# get the products and categories we are interested in
productids = set([])
categories = set([])
for e, line in enumerate( open(offers_file) ):
        if e > 0:
                li = line.strip().split(",")
                categories.add(li[1])
                productids.add(li[1]+" "+li[3]+" "+li[5])
categories = list(categories)
productids = list(productids)

category_product_unit_prices = {}
for c in categories:
        category_product_unit_prices[c] = {}

filt = open(transactions_file, "r")
filt.next()
for e, line in enumerate( filt ):
        li = line.strip().split(",")
        category = li[3]
        if category in categories:
                prodid = string.join(li[3:6]," ")
                unit = string.join(li[7:9]," ")
                if not prodid in category_product_unit_prices[category]:
                        category_product_unit_prices[category][prodid] = {}
                if not unit in category_product_unit_prices[category][prodid]:
                        category_product_unit_prices[category][prodid][unit] = []
                if float(li[10]) > 0:
                        category_product_unit_prices[category][prodid][unit].append(float(li[10]))
        if e % 10000000 == 0 and e > 0:
                print e

product_features = {}

product_comparisons = {
        '4401 105100050 13791' : {'product_container' : '19.6 OZ', 'category_container' : '19 OZ'},
        '5824 105190050 26456' : {'product_container' : '13 OZ', 'category_container' : '13 OZ'},
        '4517 105450050 1322' : {'product_container' : '18 OZ', 'category_container' : '18 OZ'}, # also mayb
h '16 OZ'
        '1726 104460040 7668' : {'product_container' : '35 CT', 'category_container' : '35 CT'},
        '9909 1089520383 28840' : {'product_container' : '5 OZ', 'category_container' : '5 OZ'},
        '5619 107717272 102504' : {'product_container' : '8 OZ', 'category_container' : '8 OZ'},
        '3509 103320030 875' : {'product_container' : '50 OZ', 'category_container' : '50 OZ'},
        '2119 108079383 6926' : {'product_container' : '144 OZ', 'category_container' : '144 OZ'},
        '9909 107127979 6732' : {'product_container' : '12 OZ', 'category_container' : '12 OZ'},
        '7205 103700030 4294' : {'product_container' : '5.8 OZ', 'category_container' : '5.8 OZ'},
        '6202 1087744888 64486' : {'product_container' : '8.8 OZ', 'category_container' : '9 OZ'},
        '5558 107120272 5072' : {'product_container' : '9.3 OZ', 'category_container' : '9 OZ'},
        '3504 104460040 7668' : {'product_container' : '22 OZ', 'category_container' : '20 OZ'},
        '5122 107106878 17311' : {'product_container' : '42.72 OZ', 'category_container' : '42.72 OZ'},
        '9115 108500080 93904' : {'product_container' : '0.75 LT', 'category_container' : '0.75 LT'},
        '2202 104460040 3718' : {'product_container' : '32 OZ', 'category_container' : '30.3 OZ'},
        '1703 104460040 7668' : {'product_container' : '32 OZ', 'category_container' : '32 OZ'},
        '706 104127141 26189' : {'product_container' : '64 OZ', 'category_container' : '64 OZ'},
        '3203 106414464 13474' : {'product_container' : '5 OZ', 'category_container' : '5 OZ'},
        '799 1076211171 17286' : {'product_container' : '12 OZ', 'category_container' : '12.7 OZ'},
        '5616 104610040 15889' : {'product_container' : '8 OZ', 'category_container' : '8 OZ'},
}

#conts = defaultdict(list)
#[[conts.__setitem__(k, conts.__getitem__(k) + category_product_unit_prices['2202'][p][k]) for k in category
['2202'][p]] for p in pids]
```

ภาพประกอบ 20 โค้ดสำหรับสร้างฟีเจอร์เปรียบเทียบราคาสินค้ากับสินค้าที่คล้ายกัน

```
for pr in productids:

        category = pr.split()[0]
        catdict = category_product_unit_prices[category]

        category_container = product_comparisons[pr]['category_container']
        product_container = product_comparisons[pr]['product_container']

        prices = []
        for prod,val in catdict.iteritems():
                if not prod == pr:
                        if category_container in val:
                                prices = prices + val[category_container]

        product_prices = catdict[pr][product_container]

        if not category_container == product_container:
                cat_size = float(category_container.split()[0])
                prod_size = float(product_container.split()[0])
                prices = np.array(prices)*prod_size/cat_size

        # compare price for our products to other products in category with same container

        # store productprice divided by median price in category
        if np.mean(prices) == 0.:
                print "mean is zero"
                import pdb;pdb.set_trace()
        mean_prod = np.mean(product_prices)/np.mean(prices)
        # store productprice divided by mean price in category
        if np.median(prices) == 0.:
                print "median is zero"
                import pdb;pdb.set_trace()
        median_prod = np.median(product_prices)/np.median(prices)
        # store which quantile it is
        median_prod_price = np.median(product_prices)
        sorted_prices = sorted(prices)
        for i, sp in enumerate(sorted_prices):
                if median_prod_price < sp:
                        quantile = float(i)/float(len(sorted_prices))
                        break
        # median price difference
        median_diff = median_prod_price-np.median(prices)

        product_features[pr] = {'median_rate' : median_prod, 'mean_rate' : mean_prod, 'quantile' : quantile, 'median_difference'
: median_diff}

of = open(os.path.join(folder,"product_cheapness_features.csv"), "w")
of.write("product_id,price_quantile,price_median_compare,price_mean_compare,price_median_difference\n")
for pr in productids:
        of.write(pr+","+str(product_features[pr]['quantile'])+","+str(product_features[pr]['median_rate'])+","+str(product_featu
res[pr]['mean_rate'])+","+str(product_features[pr]['median_difference'])+"\n")
```

ภาพประกอบ 21 โค้ดสำหรับสร้างฟีเจอร์เปรียบเทียบราคาสินค้ากับสินค้าที่คล้ายกัน (ต่อ)

10. โค้ดส่วนของการสร้างไฟล์สำหรับการส่ง Kaggle (submission file) แสดงดังภาพประกอบ 22

```
def savePredictionResultForSubmission(offer_id, algo):
    test_data = pd.io.parsers.read_csv("../data/test/all_features.csv", sep=" ")
    del test_data['repeattrips']
    del test_data['label']
    predTest = clfOneOffer[offer_id].predict_proba(test_data)
#     if features:
#         predTest = clfOneOffer[offer_id].predict_proba(test_data)
    zipall = zip(test_ids, predTest)
    result = pd.DataFrame(zipall)
    result.columns = ["id", "repeatProbability"]
    result.to_csv("../result/leave_one_offer" + str(offer_id) +"_" + algo +"_" + "_out_28_JAN_2019.csv", index=False)
    print "save result offer", offer_id, "for kaggle submission to csv file"
    print "at"+"../result/leave_one_offer" + str(offer_id) +"_" + algo +"_" + "_out_28_JAN_2019.csv"
    print "============================================="
```

ภาพประกอบ 22 โค้ดส่วนของการสร้างไฟล์สำหรับการส่ง Kaggle (submission file)

11. โค้ดส่วนของการทำ Leave One Offer Out แสดงดังภาพประกอบ 23

```
def leaveOneOfferOut(offer_id, model):
    print "leave offer =>", offer_id, "out"
    select_data[offer_id] = train_data.loc[train_data['offer_id'] != offer_id]
    select_test_data[offer_id] = train_data.loc[train_data['offer_id'] == offer_id]
    # train_label
    select_data_label[offer_id] = select_data[offer_id]['label']
    select_test_id[offer_id] = select_test_data[offer_id]['id']

    y_train = select_data[offer_id]['label']
    del select_data[offer_id]['label']
    del select_data[offer_id]['repeattrips']
    X_train = select_data[offer_id]

    y_test = select_test_data[offer_id]['label']
    del select_test_data[offer_id]['label']
    del select_test_data[offer_id]['repeattrips']
    X_test = select_test_data[offer_id]
    if model == 'xgboost':
        param = {}
        param['eval_metric'] = 'auc'
        dtrain = xgb.DMatrix(X_train, label=y_train)
        dtest = xgb.DMatrix(X_test)
        clfOneOffer[offer_id] = xgb.train(param, dtrain)
        predLeaveOfferOut[offer_id] = clfOneOffer[offer_id].predict(dtest)
    elif model == 'rfclassifier':
        # start train model
        clfOneOffer[offer_id] = RandomForestClassifier(n_estimators=200, max_depth=12,min_samples_split=10, min_sample
s_leaf=3)
        print "created rf classifier model"
        start = time.time()
        # print "train with data not offer =>", offer_id
        clfOneOffer[offer_id].fit(X_train, y_train)
        endtime = time.time() - start
        print "use time to train this offer out=>", endtime
        predLeaveOfferOut[offer_id] = clfOneOffer[offer_id].predict_proba(X_test)[:,1]
    elif model == 'xgbregressor':
        # start train model
        clfOneOffer[offer_id] = GradientBoostingRegressor(learning_rate= 0.08, max_depth= 10, n_estimators= 400)
        print "created gradient boost regressor model"
        start = time.time()
        # print "train with data not offer =>", offer_id
        clfOneOffer[offer_id].fit(X_train, y_train)
        endtime = time.time() - start
        print clf.best_params_
        print "use time to train this offer out=>", endtime
        predLeaveOfferOut[offer_id] = clfOneOffer[offer_id].predict(X_test)
    else:
        # start train model
        clfOneOffer[offer_id] = RandomForestRegressor(n_estimators=200,criterion='mse', max_depth=12,min_samples_split
=10,max_features ='sqrt', min_samples_leaf=3)
        start = time.time()
        # print "train with data not offer =>", offer_id
        clfOneOffer[offer_id].fit(X_train, y_train)
        endtime = time.time() - start
        print "use time to train this offer out=>", endtime
        predLeaveOfferOut[offer_id] = clfOneOffer[offer_id].predict(X_test)
    ## CALCULATE ROC AUC SCORE
    select_zipall[offer_id] = zip(y_test, predLeaveOfferOut[offer_id])
    resultLabelProp[offer_id] = pd.DataFrame(select_zipall[offer_id])
    resultLabelProp[offer_id].columns = ['label', 'repeatProbability']
    #print resultLabelProp[offer_id]
    resultOfferOut[offer_id] = roc_auc_score(resultLabelProp[offer_id].label, resultLabelProp[offer_id].repeatProbabil
ity)
    print "predicted leave this offer out=>", offer_id, resultOfferOut[offer_id]
```

ภาพประกอบ 23 โค้ดส่วนของการทำ Leave One Offer Out

# บรรณานุกรม


1. Kaggle. *Acquire Valued Shoppers Challenge*. 2014 3 December 2018]; Available from: https://www.kaggle.com/c/acquire-valued-shoppers-challenge.
2. Géron, A., *Hands-on machine learning with Scikit-Learn and TensorFlow: concepts, tools, and techniques to build intelligent systems*. 2017: " O'Reilly Media, Inc.".
3. Pedregosa, F.a.V., G. and Gramfort, A. and Michel, V., et al., *Scikit-learn: Machine Learning in Python.* Journal of Machine Learning Research, 2011. **12**: p. 2825--2830.
4. Chen, T.a.G., Carlos, *XGBoost: A Scalable Tree Boosting System.* 2016: p. 785--794.
5. Friedman, J., T. Hastie, and R. Tibshirani, *The elements of statistical learning*. Vol. 1. 2001: Springer series in statistics New York.
6. Philip, S., P. Shola, and A. Ovye, *Application of content-based approach in research paper recommendation system for a digital library.* International Journal of Advanced Computer Science and Applications, 2014. **5**(10).
7. Nikulin, V., *Prediction of the shoppers loyalty with aggregated data streams.* Journal of Artificial Intelligence and Soft Computing Research, 2016. **6**(2): p. 69-79.
8. Kazmi, A.H., G. Shroff, and P. Agarwal. *Generic Framework to Predict Repeat Behavior of Customers Using Their Transaction History*. in *2016 IEEE/WIC/ACM International Conference on Web Intelligence (WI)*. 2016. IEEE.
9. Auduno. *Kaggle Acquire Valued Shoppers Challenge*. 2014 2014, August 25; Available from: https://github.com/auduno/Kaggle-Acquire-Valued-Shoppers-Challenge.
10. *Repeat Buyer Prediction Competition*. 2015; Available from: https://ijcai-15.org/index.php/repeat-buyers-prediction-competition.
11. Anand, G., et al. *Deep temporal features to predict repeat buyers*. in *NIPS 2015 Workshop: Machine Learning for eCommerce*. 2015.

12. Michailidis. *How to Win a Data Science Competition: Learn from Top Kagglers*. 15 December 2018

]; Available from: https://coursera.org/lecture/competitive-data-science/acquire-valued-shoppers-challenge-part-1-nudnt.

# ประวัติผู้เขียน

| | |
|---|---|
| **ชื่อ-สกุล** | ธนัท จรณะสมบูรณ์ |
| **วัน เดือน ปี เกิด** | 21 ตุลาคม 2534 |
| **สถานที่เกิด** | กรุงเทพมหานคร |
| **วุฒิการศึกษา** | มหาวิทยาลัยศรีนครินทรวิโรฒ |