# การวิเคราะห์และทำนายค่าเป้าหมายด้วยวิธีการทาง Data Analysis
# กรณีศึกษา: การวิเคราะห์และทำนายค่ารายได้ประจำเดือนขององค์กรจากข้อมูลย้อนหลัง


**จัดทำโดย**

นายชัยพล แย้มบาน รหัสนิสิต 5931015821

**อาจารย์ที่ปรึกษา**

อ.ดร. กุลวดี ศรีพาณิชกุลชัย

ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์

จุฬาลงกรณ์มหาวิทยาลัย



รายงานเล่มนี้ใช้ประกอบการศึกษารายวิชาเอกัตศึกษาทางวิศวกรรมคอมพิวเตอร์ 2

(Individual Study In Computer Engineering 2) รหัสวิชา 2110292

ภาคเรียนที่ 2 ปีการศึกษา 2560

ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์

จุฬาลงกรณ์มหาวิทยาลัย

# บทคัดย่อ

ผู้จัดทำได้ทำการศึกษาเรื่อง การวิเคราะห์และทำนายค่าเป้าหมายด้วยวิธีการทาง Data Analysis กรณีศึกษา: การวิเคราะห์และทำนายค่ารายได้ประจำเดือนขององค์กรจากข้อมูลย้อนหลัง มีวัตถุประสงค์เพื่อเรียนรู้คำสั่งและอัลกอริทึมต่างๆ ที่ใช้ในการจัดการข้อมูล และทดลองการใช้คำสั่งเหล่านั้น เพื่อวิเคราะห์หาแบบจำลองที่สามารถนำไปใช้กับข้อมูลตัวอย่างได้ดีที่สุด

กระบวนการดำเนินงานเริ่มจากการติดตั้งซอฟต์แวร์ที่จำเป็น ได้แก่ Java, Scala และ Apache Spark จากนั้นจึงได้ศึกษาโค้ดจากปัญหาตัวอย่าง ซึ่งจะฝึกการแปลงข้อมูล สร้างตัวแปรขึ้นใหม่จากข้อมูลที่มีอยู่ สร้างฟังก์ชันในการแปลงข้อมูล สร้างแบบจำลองจากข้อมูลที่แปลงแล้วโดยใช้อัลกอริทึม Logistic Regression แล้วนำแบบจำลองดังกล่าวมาวัดประสิทธิภาพโดยการคำนวณพื้นที่ใต้กราฟ ROC ซึ่งอยู่ในระดับที่น่าพอใจ

ผู้จัดทำจึงได้ประยุกต์คำสั่งดังกล่าวกับปัญหาและข้อมูลที่ได้รับมอบหมาย โดยเขียนฟังก์ชันแปลงข้อมูลเพิ่มขึ้นมาเพื่อสร้างตัวแปรขึ้นใหม่ แล้วออกแบบการทดลอง 16 ชุด ในแต่ละชุดจะเลือกตัวแปรที่ต่างกันมาใช้ในการสร้างแบบจำลองด้วยอัลกอริทึม Linear Regression แล้ววิเคราะห์หาประสิทธิภาพของแบบจำลองที่สร้างขึ้นโดยการคำนวณค่า Root Mean Square

จากการทดลอง พบว่า แบบจำลองที่ประกอบไปด้วยตัวแปร Year, HighSeason ที่กำหนดให้เดือนช่วงเปลี่ยนผ่านเป็น 0.0, A, B และ Holidays เป็นแบบจำลองที่มีประสิทธิภาพมากที่สุดซึ่งมีค่า Root Mean Square อยู่ที่ 6.93

# กิตติกรรมประกาศ

การศึกษาในครั้งนี้จะไม่สามารถประสบความสำเร็จลุล่วงได้ หากขาดความกรุณาจาก อ.ดร.กุลวดี ศรีพาณิชกุลชัย อาจารย์ที่ปรึกษา ที่ได้ให้คำแนะนำ ชี้แจงแนวทาง ติดตาม ประเมินผล และช่วยในการแก้ไขจุดบกพร่องต่างๆมาโดยตลอด จนการศึกษานี้เสร็จสมบูรณ์ ผู้จัดทำขอขอบพระคุณเป็นอย่างสูง

ขอขอบคุณอาจารย์ และเพื่อนพี่น้อง ในภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย ที่ได้แนะนำรายวิชาเอกัตศึกษาทางวิศวกรรมคอมพิวเตอร์ 2 (Individual Study In Computer Engineering 2) รหัสวิชา 2110292 และได้ให้โอกาสในการพัฒนาองค์ความรู้ในด้านที่สนใจ สามารถต่อยอดและนำไปใช้ในการทำงานได้ในอนาคต

ท้ายที่สุดนี้ หากการศึกษาหรือรายงานเล่มนี้มีข้อบกพร่องประการใด ผู้จัดทำขออภัยมา ณ ที่นี้ และหวังว่า การศึกษานี้จะเป็นประโยชน์ต่อผู้อ่าน หรือเป็นแรงจูงใจให้ผู้อ่านมีความสนใจที่จะศึกษาเกี่ยวกับวิทยาศาสตร์ข้อมูลมากขึ้น

ชัยพล แย้มบาน

# สารบัญ

# สารบัญ (ต่อ)

# สารบัญ (ต่อ)

# สารบัญตาราง

# บทที่ 1

# บทนำ

## 1.1 ความเป็นมา

ปัจจุบัน มีข้อมูลมากมายถูกเก็บไว้ในระบบคอมพิวเตอร์เป็นจำนวนมาก ข้อมูลเหล่านั้นมีศักยภาพที่สามารถนำมาวิเคราะห์เพื่อหาความสัมพันธ์ระหว่างข้อมูล และทำนายค่าข้อมูลที่ต้องการได้ การเก็บข้อมูลจึงมีความสำคัญมากโดยเฉพาะในด้านธุรกิจ เพราะสามารถคาดการณ์ผลที่จะเกิดขึ้นได้จากข้อมูลของลูกค้าหรือผู้ใช้งาน ทำให้สามารถเข้าถึงกลุ่มเป้าหมายได้โดยง่ายและตรงจุด ตอบโจทย์ที่ลูกค้าต้องการได้อย่างรวดเร็ว ดึงดูดลูกค้าได้มากขึ้น นอกจากนั้น ข้อมูลผลประกอบการที่ผ่านๆมา ก็สามารถเป็นข้อมูลที่นำมาคาดการณ์ผลประกอบการในอนาคตได้ด้วย ทำให้ช่วยในการกำหนดนโยบายต่างๆ หรือหาวิธีการแก้ปัญหาหนึ่งๆ ที่คุ้มค่าที่สุด จนเกิดศาสตร์ที่เรียกว่า วิทยาศาสตร์ข้อมูล (Data Science) ขึ้น

ภาคธุรกิจขนาดใหญ่ทั่วโลก เริ่มมีตำแหน่งงานที่ทำงานเกี่ยวกับการวิเคราะห์และทำนายข้อมูลโดยเฉพาะ เช่น นักวิทยาศาสตร์ข้อมูล(Data Scientist) นักวิเคราะห์ข้อมูล(Data Analyst) เป็นต้น ตำแหน่งเหล่านี้จะเป็นที่ต้องการมากขึ้นเรื่อยๆ ถึงแม้ในประเทศไทยจะยังไม่พบเห็นมากนัก อย่างไรก็ตาม ผู้จัดทำจึงมีความสนใจที่จะศึกษาและทดลองวิเคราะห์ข้อมูลชุดหนึ่ง หาความสัมพันธ์ของข้อมูล และทำนายผลลัพธ์จากข้อมูลชุดดังกล่าว เพื่อเป็นแนวทางในการต่อยอดองค์ความรู้ต่อไป

## 1.2 วัตถุประสงค์

- ศึกษาการใช้โปรแกรม และคำสั่งต่างๆที่ใช้ในการดำเนินงาน
- ศึกษาอัลกอริทึมที่ใช้ในการวิเคราะห์และทำนายข้อมูล
- ฝึกการวิเคราะห์ เลือกใช้ตัวแปรเพื่อสร้างแบบจำลองที่ใช้ทำนายข้อมูล

## 1.3 ขอบเขตของการดำเนินงาน

ในการศึกษานี้ ผู้จัดทำศึกษา ติดตั้งและฝึกใช้โปรแกรมที่ใช้ในการดำเนินงาน ทดลองเลือกกลุ่มตัวแปรจากตัวแปรในรูปแบบต่างๆ สร้างแบบทำนายของแต่ละรูปแบบ วิเคราะห์ความผิดพลาดและเลือกรูปแบบที่ดีที่สุด ภายในเวลา 1 ภาคการศึกษา

# บทที่ 2

# ความรู้และทฤษฎี

## 2.1 เนื้อหาทั่วไป

### Machine Learning

Machine Learning มีคำแปลที่ตรงตัวว่า “การเรียนรู้ของเครื่องจักร” ซึ่งหมายถึง การทำให้ระบบคอมพิวเตอร์เรียนรู้ความสัมพันธ์ของข้อมูลด้วยตัวเอง ผู้ใช้จะป้อนเพียงข้อมูลและผลลัพธ์เท่านั้น คอมพิวเตอร์จะสร้างแบบจำลองหรือสมการที่แสดงความความสัมพันธ์ของข้อมูลและผลลัพธ์ดังกล่าวด้วยตนเอง โดยที่ผู้ใช้ไม่ต้องป้อนวิธีคิดเพื่อหาผลลัพธ์นั้นๆ ซึ่งยิ่งจำนวนชุดข้อมูลมาก แบบจำลองจะยิ่งถูกต้องมากขึ้น ผลลัพธ์จะแม่นยำขึ้น

### Linear Regression

เป็นแบบจำลองอย่างหนึ่งใน Machine Learning เป็นการเรียนรู้ความสัมพันธ์ของข้อมูลในลักษณะของการจำลองเป็นสมการเชิงเส้นระหว่างผลลัพธ์และตัวแปรต่างๆของข้อมูล สำหรับการเรียนรู้แบบ Linear Regression นั้นตัวแปรทั้งหมดรวมไปถึงผลลัพธ์จะเป็นข้อมูลเชิงปริมาณ มักจะใช้ในปัญหาเช่น การทำนายผลประกอบการของบริษัท การทำนายจำนวนประชากรในเมืองๆหนึ่ง เป็นต้น สมการของแบบจำลองนี้[1] คือ

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_i x_i + \varepsilon$$

โดยที่ y คือ ผลลัพธ์ของข้อมูล

x คือ ค่าข้อมูลของแต่ละตัวแปร

$\varepsilon$ คือ ค่าความผิดพลาด

$\beta$ คือ ค่าสัมประสิทธิ์ ซึ่งจะได้จากการสร้างแบบจำลอง

### Logistic Regression

เป็นแบบจำลองอย่างหนึ่งใน Machine Learning ลักษณะการทำงานและสมการจะคล้ายกับ Linear Regression แต่ผลลัพธ์จะเป็นค่าในช่วง [0,1] มักจะใช้ทำนายโอกาสในการเกิดเหตุการณ์หนึ่งๆ เช่น โอกาสที่จะรักษาโรคหนึ่งให้หาย เป็นต้น

## Apache Spark


เป็นเครื่องมือหนึ่งที่ใช้ในการวเคราะห์ข้อมูล Big Data โดยจะต้องมี Server ที่ต้องติดตั้งและให้บริการ มีโครงสร้างและรูปแบบการเก็บข้อมูล และมีภาษาที่เขียนเพื่อวิเคราะห์หรือจัดการข้อมูล เช่น Java, Python, Scala เป็นต้น

Spark มีการทำงานที่รวดเร็วกว่า Hadoop และใช้ร่วมกับ Hadoop ได้ และมีจุดเด่นคือถนัดในด้านการดึงข้อมูล คัดกรอง และนำไปใช้งานต่อ (Extract, Load, Transform) ซึ่งในการศึกษาครั้งนี้จะมีการสั่งงานตามกระบวนการดังกล่าว

## Scala

เป็นภาษาหนึ่งที่ใช้ในการเขียนคำสั่งเพื่อวิเคราะห์ จัดการ และประมวลผลข้อมูล เป็นภาษาพี่น้องกับ Java ทำงานอยู่บน Java Virtual Machine และมีคำสั่งที่สั้นกระชับกว่า Java

## Dataframe

เป็นโครงสร้างข้อมูล ที่มีลักษณะคล้ายตาราง โดยคอลัมน์จะแสดงค่าตัวแปร ส่วนแถวจะแสดงลำดับของชุดข้อมูล

| | sepal length (cm) | sepal width (cm) | petal length (cm) | petal width (cm) |
|---|---|---|---|---|
| **0** | 5.1 | 3.5 | 1.4 | 0.2 |
| **1** | 4.9 | 3.0 | 1.4 | 0.2 |
| **2** | 4.7 | 3.2 | 1.3 | 0.2 |
| **3** | 4.6 | 3.1 | 1.5 | 0.2 |
| **4** | 5.0 | 3.6 | 1.4 | 0.2 |
| **5** | 5.4 | 3.9 | 1.7 | 0.4 |
| **6** | 4.6 | 3.4 | 1.4 | 0.3 |
| **7** | 5.0 | 3.4 | 1.5 | 0.2 |
| **8** | 4.4 | 2.9 | 1.4 | 0.2 |
| **9** | 4.9 | 3.1 | 1.5 | 0.1 |

ตารางที่ 1 : ตัวอย่าง Dataframe[2]

## 2.2 ขั้นตอนการทำงาน

**Transformer**

เป็นตัวแปลง (Transform) ข้อมูลดิบเป็นข้อมูลค่าอื่น หรืออาจเป็นข้อมูลประเภทอื่น เช่น การแบ่งกลุ่ม การเปลี่ยนสตริงเป็นจำนวน เป็นต้น สำหรับการศึกษานี้จะมี Transformer อยู่หลายชนิด

**Estimator**

มีลักษณะคล้ายกับ Transformer แต่จะมีการ Fit (ปรับข้อมูลหรือ Model ให้เข้ากัน) ก่อนที่จะ Transform

**Bucketizer**

เป็น Transformer ที่ทำหน้าที่จัดกลุ่มจำนวนจริงได้ตั้งแต่ -∞ ถึง ∞ เป็นช่วงๆ โดยจะมีค่าที่เป็นตัวแบ่งหลายๆ ค่า ซึ่งแบ่งค่าออกเป็นหลายๆช่วง ผลลัพธ์จาก Bucketizer จะเป็นหมายเลขประจำช่วง โดยจะเริ่มจากช่วงของค่าต่ำที่สุดเป็น 0.0 ช่วงต่อไปเป็น 1.0 เรียงตามลำดับ หากต้องการครอบคลุมจำนวนจริงทั้งหมด จะต้องใส่ค่า Double.NegativeInfinity (-∞) และ Double.PositiveInfinity (∞) ไว้ที่หัวและท้ายของตัวแบ่ง ตามลำดับด้วย

**ตัวอย่าง** : หากค่าที่ใส่เป็น 0, 10, 20 ข้อมูลจะแบ่งช่วงเป็น (0,10) และ (10,20)

หากค่าที่ใส่เป็น Double.NegativeInfinity, 0, Double.PositiveInfinity ข้อมูลจะแบ่งช่วงเป็น (-∞,0) และ (0, ∞) เป็นต้น

**String Indexer**

เป็น Transformer ที่ทำหน้าที่เปลี่ยนสตริงเป็นจำนวนจริง โดยสตริงที่มีความถี่มากที่สุด จะเปลี่ยนเป็น 0.0 สตริงที่มีความถี่รองลงมาเป็น 1.0, 2.0, 3.0, ... ตามลำดับ

**Vector Assembler**

เป็น Transformer ที่รวมข้อมูลหลายๆตัวแปรเข้าเป็นเวกเตอร์ ซึ่งจะเอาเวกเตอร์ที่ได้จากการรวมนี้ไปใช้สร้าง Model ต่อไป

**Normalizer**

เป็น Transformer ที่ปรับขนาดของเวกเตอร์ให้เป็นเวกเตอร์หนึ่งหน่วย (Unit Vector) ที่มีทิศทางเดียวกับเวกเตอร์เดิม

ขนาดของเวกเตอร์คำนวณได้จากสมการ[3]

$$\|\mathbf{x}\|_p := \left( \sum_{i=1}^{n} |x_i|^p \right)^{1/p}$$

โดยในการคำนวณขนาดเวกเตอร์ทั่วไป จะใช้ค่า p = 2 แต่สามารถตั้งค่าให้ p เป็นค่าอื่นได้

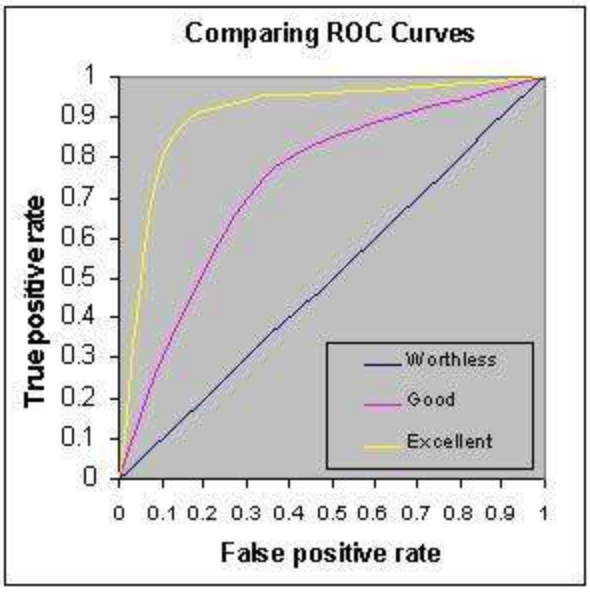
## 2.3 สถิติที่ใช้

**กราฟ ROC**

ROC ย่อมาจาก Receiver Operating Characteristic กราฟ ROC เป็นกราฟที่ใช้วัดประสิทธิภาพของการจำแนกข้อมูลที่มีผลเป็น binary (1/0) เช่น การคาดการณ์ว่าผู้ป่วยจะเสียชีวิตหรือไม่ เป็นต้น

ในการศึกษานี้ กราฟนี้จะใช้กับ Logistic Regression ที่ทำนายว่าผู้โดยสารบนเรือ Titanic จะรอดหรือไม่ โดยทดลองเลือกตัวแบ่งค่าโอกาสตั้งแต่ 0 ถึง 1 (หากโอกาสต่ำกว่าตัวแบ่ง ถือว่าทำนายไว้ว่าไม่รอด แต่ถ้าโอกาสสูงกว่าตัวแบ่ง จะถือว่ารอด สำหรับตัวแบ่งแต่ละตัว จะวัดอัตรา True Positive Rate (สัดส่วนจำนวนคนที่รอดตามที่ทำนายต่อจำนวนคนที่รอดทั้งหมด) และ False Positive Rate (สัดส่วนจำนวนคนที่ไม่รอดแต่ถูกทายว่ารอดต่อจำนวนคนที่ไม่รอดทั้งหมด) นำค่าทั้งสองจากตัวแบ่งทั้งหมดมา plot ในกราฟ โดยให้ True Positive Rate เป็นแกน Y และ False Positive Rate เป็นแกน X จนเป็นเส้นโค้ง พื้นที่ใต้เส้นโค้งนั้นเรียกว่า “พื้นที่ใต้ ROC”

ยิ่งพื้นที่นี้ใกล้ 1 มาก แสดงว่า model มีประสิทธิภาพมาก




**รากที่สองของกำลังสองเฉลี่ย (Root Mean Square)**

เป็นสมการที่นิยมใช้ในการคำนวณความผิดพลาดของการทดลองทางสถิติและวิทยาศาสตร์ มีสมการดังนี้[5]

$$x_{\mathrm{rms}} = \sqrt{\frac{1}{N}\sum_{i=1}^{N} x_i^2} = \sqrt{\frac{x_1^2 + x_2^2 + \cdots + x_N^2}{N}}$$

โดยที่ N เป็นจำนวนข้อมูล

# บทที่ 3

# การดำเนินงาน

## 3.1 ข้อมูลทั่วไป

เวอร์ชั่นของ Spark : 2.2.1

เวอร์ชั่นของ Scala : 2.11.8

ระบบปฏิบัติการที่ใช้ : Ubuntu 16.04 LTS

## 3.2 ขั้นตอนการดำเนินงาน

การดำเนินงานทั้งหมด จะทำภายใต้ระบบปฏิบัติการ Ubuntu 16.04 LTS ใน Virtual Machine โดยได้มีการติดตั้ง Spark และ Scala ลงไปแล้ว สำหรับขั้นตอนการใช้งานมีดังนี้

**ขั้นศึกษา**

ผู้จัดทำได้ทำตามปฏิบัติการตามปัญหาตัวอย่าง[6] และโค้ดตัวอย่าง[7] ในเว็บไซต์ ซึ่งเป็นปัญหาเกี่ยวกับการทำนายว่าผู้โดยสารจะรอดจากเรือไททานิกได้หรือไม่ โดยมีข้อมูลต่างๆ ซึ่งมีขั้นตอนโดยรวมดังนี้

1. เปิด Terminal

2. เริ่มต้น Spark standalone master server โดยให้เข้าไปที่โฟลเดอร์ที่ดาวน์โหลด spark ไว้ (ของผู้จัดทำเป็น nonpro@nonpro-VirtualBox:~/Downloads/spark-2.2.1-bin-hadoop2.7) แล้วใช้คำสั่ง

```
./sbin/start-master.sh
```

เมื่อเปิดแล้วจะมี URL ขึ้นมา สามารถดูได้จาก Web UI ของ Master (โดยปกติจะเป็นเว็บ http://localhost:8080 สำหรับ URL ของผู้จัดทำจะเป็น spark://nonpro-VirtualBox:7077

3. เริ่มต้น Spark shell ด้วยคำสั่ง

```
spark-shell --master spark://yourspark-server-url --packages
com.databricks:spark-csv_2.11:1.3.0
```

โดยให้แทนที่ spark://yourspark-server-url ด้วย URL ที่ได้ในข้อ 2

4. Import class ที่จำเป็นในการใช้งานซึ่งอยู่ในโค้ดตัวอย่าง

5. เริ่มต้นโหลดไฟล์ข้อมูล (.csv)[8] ลงไปในโปรแกรม โดยตั้งฟังก์ชันโหลดขึ้นมาใหม่ด้วยคำสั่งดังนี้

```
def load(path: String, sqlContext: SQLContext, featuresArr: String*):
DataFrame = {
    var data = sqlContext.read.format("com.databricks.spark.csv")
  .option("header", "true")
  .option("inferSchema", "true")
  .load(path)
  .toDF(featuresArr: _*)
    return data
  }
```

ฟังก์ชันนี้มีพารามิเตอร์ 3 ตัว คือ path ของไฟล์ที่จะอัพโหลด, sqlContext และ featureArr ซึ่งจะประกอบไปด้วยคอลัมน์ของข้อมูล

สร้าง sqlContext จากคำสั่ง

```
val spark = SparkSession.builder().master("spark://yourspark-server-url
").appName("Titanic").getOrCreate()
    val sc = spark.sparkContext
    val sqlContext = spark.sqlContext
```

โดยให้แทนที่ spark://yourspark-server-url ด้วย URL ที่ได้ในข้อ 2

โหลดไฟล์ข้อมูลลงไปใน train_data ด้วยคำสั่ง

```
var train_data = load("/kaggle/titanic/train.csv",
      sqlContext,
      "PassengerId", "Survived", "Pclass", "Name", "Sex", "Age", "SibSp",
"Parch", "Ticket", "Fare", "Cabin", "Embarked"
      ).cache()
```

เราสามารถดูผังข้อมูลของไฟล์ได้ด้วยคำสั่ง

```
train_data.printSchema()
```

ซึ่งจะขึ้นผลลัพธ์ดังนี้

```
root
 |-- PassengerId: integer (nullable = true)
 |-- Survived: integer (nullable = true)
 |-- Pclass: integer (nullable = true)
 |-- Name: string (nullable = true)
 |-- Sex: string (nullable = true)
 |-- Age: double (nullable = true)
 |-- SibSp: integer (nullable = true)
 |-- Parch: integer (nullable = true)
 |-- Ticket: string (nullable = true)
 |-- Fare: double (nullable = true)
 |-- Cabin: string (nullable = true)
 |-- Embarked: string (nullable = true)
```

สามารถสั่งให้แสดงตารางได้ด้วยคำสั่ง

```
train_data.show()
```

ซึ่งจะขึ้นผลลัพธ์เป็นตารางข้อมูลทั้งหมด

สามารถสั่งให้แสดงค่าทางสถิติของคอลัมน์ที่เป็นค่าจำนวนได้ด้วยคำสั่ง

```
train_data.describe("Fare").show()
```

ซึ่งจะขึ้นผลลัพธ์

```
+-------+------------------+
|summary|              Fare|
+-------+------------------+
|  count|               891|
|   mean|32.204207968574615|
| stddev|49.665534444774124|
|    min|                 0|
|    max|              93.5|
+-------+------------------+
```

6. ทำการ Pre-process ซึ่งจะเป็นการจัดการข้อมูลก่อนนำไปวิเคราะห์ เพราะข้อมูลที่มีอยู่ ไม่อยู่ในรูปแบบที่สามารถนำไปวิเคราะห์ได้โดยตรง เช่น เป็นจำนวนเต็ม เป็น string หรือมีข้อมูลขาดหายไป เป็นต้น ซึ่งจะต้องปรับให้เป็นจำนวนจริง มีข้อมูลครบทุกช่อง

ในไฟล์ข้อมูลยังมีจุดที่ผิดปกติอยู่ที่คอลัมน์ Age ซึ่งบางแถวยังไม่มีข้อมูลอายุ จัดการโดยหาค่าเฉลี่ยของอายุของทุกแถวที่มีไปเก็บไว้ใน avgAge แล้วเอาค่าดังกล่าวไปเติมลงในทุกแถวที่ขาด

```
var avgAge = train_data.select(mean("Age")).first()(0)
.asInstanceOf[Double]
train_data = train_data.na.fill(avgAge, Seq("Age"))
```

7. ข้อมูลแต่ละคอลัมน์สามารถนำมาใช้ร่วมกับคอลัมน์อื่นซึ่งสร้างคอลัมน์ใหม่ ซึ่งอาจนำไปใช้แล้วทำให้วิเคราะห์ได้ดีขึ้น วิธีการนี้เรียกว่า Feature Engineering คอลัมน์ที่นำไปใช้วิเคราะห์ เรียกว่า Feature ในตัวอย่างได้มีการสร้างคอลัมน์ขึ้นมาใหม่ ซึ่งจะต้องสร้างฟังก์ชันที่ใช้แปลงค่า ตัวอย่างเช่นฟังก์ชัน findtitle ที่นำข้อมูลชื่อแปลงเป็นคำนำหน้าชื่อ มีคำสั่งดังนี้

```
val findTitle = sqlContext.udf.register("findTitle", (name: String) => {
      val pattern =
"(Dr|Mrs?|Ms|Miss|Master|Rev|Capt|Mlle|Col|Major|Sir|Lady|Mme|Don)\\.".r
      val matchedStr = pattern.findFirstIn(name)
      var title = matchedStr match {
        case Some(s) => matchedStr.getOrElse("Other.")
        case None => "Other."
      }
      if (title.equals("Don.") || title.equals("Major.") ||
title.equals("Capt."))
        title = "Sir."
      if (title.equals("Mlle.") || title.equals("Mme."))
          title = "Miss."
      title
    })
```

และสั่งเพิ่มคอลัมน์ที่เป็นผลลัพธ์มาจากฟังก์ชัน findtitle ด้วยคำสั่ง

```
train_data = train_data.withColumn("Title", findTitle(train_data("Name")))
```

ในโค้ดตัวอย่าง ได้มีการสร้างคอลัมน์เพิ่มจากการสร้างฟังก์ชันเองได้แก่ Sex, Title, PClass, Family, Survived

8. ใช้ Transformer จาก library ได้แก่

สร้าง SexIndex ด้วย StringIndexer ใช้คำสั่งดังนี้

```
val sexInd = new StringIndexer().setInputCol("Sex")
.setOutputCol("SexIndex")
```

และสร้าง TitleIndex ด้วย StringIndexer เช่นกัน

สร้าง FareBucketed (แบ่งค่าโดยสารเป็นช่วงๆ) ด้วย Bucketizer ซึ่งต้องสร้าง array ที่เป็นตัวแบ่ง(ในที่นี้คือ fareSplits) สร้างฟังก์ชันด้วยคำสั่งดังนี้

```
val fareSplits = Array(0.0,10.0,20.0,30.0,40.0,Double.PositiveInfinity)
val fareBucketize = new Bucketizer().setInputCol("Fare")
.setOutputCol("FareBucketed").setSplits(fareSplits)
```

รวม Feature เป็น Vector ด้วย VectorAssembler สร้างฟังก์ชันด้วยคำสั่งดังนี้

```
val assembler = new VectorAssembler().setInputCols(Array("SexIndex",
"Age", "TitleIndex", "Pclass",
"Family","FareBucketed")).setOutputCol("features_temp")
```

ซึ่งจะสร้างคอลัมน์ผลลัพธ์คือ features_temp

ใช้ Normalizer ย่อขนาดเวกเตอร์ใน features_temp ด้วยคำสั่งดังนี้

```
val normalizer = new Normalizer().setInputCol("features_temp")
.setOutputCol("features").setP(1.0)
```

สร้างคอลัมน์ผลลัพธ์คือ features ส่วนค่าพารามิเตอร์ใน setP คือค่า p ในสมการหาขนาดของเวกเตอร์ที่อยู่ในข้อมูลของ Vector Assembler ในบทที่ 2

หมายเหตุ: ขั้นตอนนี้เป็นเพียงการสร้างฟังก์ชันแปลงข้อมูลด้วย Transformer เท่านั้น ยังไม่มีการนำฟังก์ชันดังกล่าวไปใช้กับ train_data และยังไม่เกิดคอลัมน์จากฟังก์ชันดังกล่าวขึ้นแต่อย่างใด

9. สร้าง Model โดยใช้ Logistic Regression Algorithm โดยสร้าง LogisticRegression Component (เป็น Estimator มีการ fit ก่อนจะ transform) ด้วยคำสั่ง

```
val lr = new LogisticRegression().setMaxIter(10)
lr.setLabelCol("Survived")
```

สร้าง Pipeline (Array ของ Estimator และ Transformer ที่ใช้) ด้วยคำสั่ง

```
val pipeline = new Pipeline().setStages(Array(sexInd, titleInd,
fareBucketize, assembler, normalizer,lr))
```

ในตัวอย่างมีการแบ่ง train_data ออกเป็น 2 ส่วน เป็น train (80% ใช้ให้ model เรียนรู้) กับ test (20% ใช้ทดสอบ) ด้วยคำสั่ง

```
val splits = train_data.randomSplit(Array(0.8, 0.2), seed = 11L)
val train = splits(0).cache()
val test = splits(1).cache()
```

จากนั้นเอา pipeline ไป fit กับ train ซึ่งจะได้ model (เป็น transformer) และเอา model ไปใช้กับ test

```
var model = pipeline.fit(train)
var result = model.transform(test)
```

นำ result มาทดสอบประสิทธิภาพของ model โดยเลือกคอลัมน์ prediction และ survived นำมาหาพื้นที่ใต้กราฟ ROC

```
result = result.select("prediction","Survived")
val predictionAndLabels = result.map { row =>
      (row.get(0).asInstanceOf[Double],row.get(1).asInstanceOf[Double])
    }
val metrics = new BinaryClassificationMetrics(predictionAndLabels)
println("Area under ROC = " + metrics.areaUnderROC())
```

10. นำ model ไปใช้ทำนายข้อมูลอื่น โดยให้ model เรียนรู้ข้อมูลใน train_data ทั้งหมดก่อน

```
model = pipeline.fit(train_data)
```

โหลดไฟล์ submission_data แล้วให้ผ่านขั้นตอนที่ 6, 7 และ 9 โดยไม่ต้องสร้างฟังก์ชันใหม่ แบ่งข้อมูลหรือวิเคราะห์หาพื้นที่ใต้กราฟ ROC อีกครั้ง ผลลัพธ์ที่ได้จะมีผลการทำนายจาก model นี้อยู่ด้วย

**ขั้นปฏิบัติ**

ปัญหาในขั้นปฏิบัตินี้ เป็นโจทย์ที่ให้ทำนายรายได้ขององค์กรเป็นรายเดือน เมื่อมีข้อมูลเป็นค่า A, B, C, Month, Year ของทุกปี และผลลัพธ์(Target) ของปีที่ผ่านๆมา โดยใช้ข้อมูลของปี 2014 และ 2015 เป็นข้อมูลให้ model เรียนรู้ และใช้ข้อมูลของปี 2016 เป็นข้อมูลที่ทดสอบความผิดพลาด

ขั้นตอนการวิเคราะห์จะคล้ายกับขั้นศึกษา แต่จะมีจุดแตกต่างดังนี้

1. เนื่องจากค่าเป้าหมายเป็นรายได้ ซึ่งไม่เป็น binary จึงไม่ใช้ Logistic Regression แต่ใช้ Linear Regression แทน ซึ่งมีสมการการคำนวณที่คล้ายคลึงกัน

2. ต้อง import Linear Regression เพิ่มขึ้นมา ด้วยคำสั่ง

```
import org.apache.spark.ml.regression.LinearRegression
```

3. การวิเคราะห์ประสิทธิภาพ ใช้วิธีหา Root Mean Square ของร้อยละของความผิดพลาดของ test data แต่ละชุด โดยร้อยละของความผิดพลาดจะคำนวณจากสมการ

$$\text{ร้อยละของความผิดพลาด} = \frac{\text{รายได้ที่ทำนายได้} - \text{รายได้ที่แท้จริง}}{\text{รายได้ที่แท้จริง}}$$

โดยจะสร้างคอลัมน์ Error% เก็บค่าร้อยละของความผิดพลาด และ Err%sq เป็นค่ากำลังสองของ Error% ขึ้นมา และใช้ mean ของ Err%sq มาหารากที่สอง เป็นค่าที่นำมาใช้วิเคราะห์และเปรียบเทียบ

4. ไม่ต้องใช้ Transformer บางตัวที่เคยใช้ในขั้นศึกษา เช่น StringIndexer, Bucketizer แต่มีการสร้างฟังก์ชันขึ้นมาเองเพื่อสร้างคอลัมน์ใหม่ที่จะนำมาใช้งาน คอลัมน์ที่เพิ่มเติมขึ้นมามีดังนี้

- HighSeason : นำค่า Month มาจำแนกว่าเป็นฤดูท่องเที่ยวหรือไม่ โดย

เดือนธันวาคม ถึง เดือนมีนาคม เป็น High Season (ให้ค่า 1.0)

เดือนพฤษภาคม ถึง เดือนตุลาคม เป็น Low Season (ให้ค่า 0.0)

เดือนเมษายน และเดือนพฤศจิกายน เป็นช่วงเปลี่ยนผ่าน (ทดลองให้ค่าทั้ง 1.0, 0.5 และ 0.0)

- DaysOfMonth : นำค่า Month และ Year มาหาเป็นจำนวนวันในเดือนนั้น

- Weekends : นำค่า Month, Year และ DaysOfMonth มาหาจำนวนวันเสาร์-อาทิตย์ของเดือนนั้น

- Holidays : นำค่า Month มาหาจำนวนวันหยุดนักขัตฤกษ์ของเดือนนั้น หากวันหยุดดังกล่าวตรงกับวันเสาร์-อาทิตย์ ให้ถือว่ามีวันหยุดชดเชยในเดือนนั้นและนับวันหยุดชดเชยนั้นแทน วันหยุดนักขัตฤกษ์แต่ละเดือนที่นำมาพิจารณา มีดังนี้

เดือนมกราคม : วันขึ้นปีใหม่

เดือนกุมภาพันธ์ : วันมาฆบูชา

เดือนเมษายน : วันจักรี และวันสงกรานต์(3 วัน)

เดือนพฤษภาคม : วันแรงงาน วันฉัตรมงคล และวันวิสาขบูชา

เดือนกรกฎาคม : วันอาสาฬหบูชา และวันเข้าพรรษา

เดือนสิงหาคม : วันแม่แห่งชาติ

เดือนธันวาคม : วันพ่อแห่งชาติ วันรัฐธรรมนูญ และวันสิ้นปี

- YearM2000 : นำค่า Year มาลบด้วย 2000
- Yearsqrt : นำค่า Year มาถอดรากที่สอง
- Weekdays : นำค่า DaysOfMonth, Weekends และ Holidays มาหาวันที่ไม่ใช่วันหยุดทั้งหมด

ผู้จัดทำได้ทดลองเลือกคอลัมน์ต่างๆ มาเป็น Features หลากหลายรูปแบบ เพื่อนำมาวิเคราะห์ประสิทธิภาพดังนี้

| การทดลองที่ | Features |
|---|---|
| 1 | A, B, C |
| 2 | A, B |
| 3 | Year, A, B |
| 4 | Year, A, B, C |
| 5 | Year, Month, A, B, C |
| 6 | Year, Month, A. B |
| 7 | Year, HighSeason(0.5), A, B |
| 8 | Year, HighSeason(0.0), A, B |
| 9 | Year, HighSeason(1.0), A, B |
| 10 | Year, HighSeason(0.0), A, B, DaysOfMonth |
| 11 | Year, HighSeason(0.0), A, B, Weekends |
| 12 | Year, HighSeason(0.0), A, B, Holidays |
| 13 | YearM2000, HighSeason(0.0), A, B, Holidays |
| 14 | Yearsqrt, HighSeason(0.0), A, B, Holidays |
| 15 | Year, HighSeason(0.0), A, B, DaysOfMonth, Weekends, Holidays |
| 16 | Year, HighSeason(0.0), A, B, Weekdays |

ตารางที่ 2 : กลุ่ม Feature ที่ใช้ในการทดลองต่างๆ

หมายเหตุ HighSeason(x) หมายถึง HighSeason ที่ให้ค่าของเดือนช่วงเปลี่ยนผ่านเป็น x

# บทที่ 4

# ผลการดำเนินงาน

ผลการทดลองเลือก Features จากบทที่ 3 ให้ผลดังนี้

| การทดลองที่ | Features | Root Mean Square ของร้อยละความผิดพลาด |
|---|---|---|
| 1 | A, B, C | 18.12 |
| 2 | A, B | 16.37 |
| 3 | Year, A, B | 8.09 |
| 4 | Year, A, B, C | 9.98 |
| 5 | Year, Month, A, B, C | 12.38 |
| 6 | Year, Month, A. B | 9.04 |
| 7 | Year, HighSeason(0.5), A, B | 7.42 |
| 8 | Year, HighSeason(0.0), A, B | 7.14 |
| 9 | Year, HighSeason(1.0), A, B | 7.45 |
| 10 | Year, HighSeason(0.0), A, B, DaysOfMonth | 7.38 |
| 11 | Year, HighSeason(0.0), A, B, Weekends | 7.38 |
| 12 | Year, HighSeason(0.0), A, B, Holidays | 6.93 |
| 13 | YearM2000, HighSeason(0.0), A, B, Holidays | 11.99 |
| 14 | Yearsqrt, HighSeason(0.0), A, B, Holidays | 7.29 |
| 15 | Year, HighSeason(0.0), A, B, DaysOfMonth, Weekends, Holidays | 7.47 |
| 16 | Year, HighSeason(0,0), A, B, Weekdays | 6.94 |

ตารางที่ 3 : ผลการทดลองจากกลุ่ม Feature ต่างๆ

**อภิปรายผลการทดลอง**

จากการทดลองที่ 1 ถึง 4 พบว่า การเพิ่ม Year เข้ามาใน features จะทำให้ความผิดพลาดลดลง ดังนั้นจึงใส่ Year ในการทดลองที่ 5 เป็นต้นไป

จากการทดลองที่ 3 ถึง 6 พบว่า การเพิ่ม Month ทำให้ความผิดพลาดเพิ่มขึ้น จึงเอาออก

จากการทดลองที่ 1 ถึง 6 พบว่า การเพิ่ม C จะเกิดความผิดพลาดมากกว่ากรณีที่ไม่มี จึงเอาออก

จากการทดลองที่ 3, 7, 8 และ 9 พบว่า การเพิ่ม HighSeason ทำให้ได้ผลดีขึ้น โดยดีที่สุดเมื่อให้เดือนที่เปลี่ยนผ่านเป็น 0.0

จากการทดลองที่ 8, 10, 11 และ 12 พบว่า การเพิ่ม Holidays ช่วยให้ผลดีขึ้น แต่ DaysOfMonth และ Weekends ไม่ได้ช่วยมากนัก

จากการทดลองที่ 12 ถึง 14 พบว่า การใช้ Year โดยตรงจะได้ผลดีกว่า YearM2000 และ Yearsqrt

จากการทดลองที่ 12, 15 และ 16 พบว่า ประเภทของวันที่ทำให้ผลดีที่สุด คือ Holidays ซึ่งใกล้เคียงกับ Weekdays

จากทั้ง 16 การทดลอง Features ที่ดีที่สุด คือ Feature ที่ประกอบไปด้วย Year, HighSeason(0.0), A, B และ Holidays โดยมี Root Mean Square ของร้อยละความผิดพลาดอยู่ที่ 6.93

# บทที่ 5

# สรุปผล และข้อเสนอแนะ

## 5.1 สรุปผลการศึกษา

จากผลการศึกษา การทดลองที่ 12 ได้ผลดีที่สุด โดย Feature ที่เลือกประกอบไปด้วย Year, HighSeason ที่กำหนดให้เดือนช่วงเปลี่ยนผ่านเป็น 0.0, A, B และ Holidays และมีค่า Root Mean Square ของร้อยละความผิดพลาดอยู่ที่ 6.93 ทำให้สามารถคาดเดาได้ว่า ค่าปี สถานะฤดูท่องเที่ยว ค่า A, B และจำนวนวันหยุดนักขัตฤกษ์ เป็นปัจจัยที่สำคัญที่มีผลต่อรายได้ขององค์กร

## 5.2 ข้อเสนอแนะ

- จำนวนข้อมูลมีน้อยเกินไป (มีเพียง 24 แถว 5 คอลัมน์) ทำให้ผลการทำนายยังไม่แม่นยำพอ ควรจะเพิ่มข้อมูลให้มากขึ้น เพื่อให้ทำนายได้แม่นยำขึ้น จนสามารถลดค่า Root Mean Square ของร้อยละความผิดพลาดลงเหลือต่ำกว่า 5 ได้

- ผู้จัดทำสังเกตว่า ค่าที่ทำนายไว้นั้น เกือบทั้งหมดจะต่ำกว่าค่าจริง (Under predicted) ส่วนค่าที่เกิน จะเกินไม่มาก เพียง 2–3% เท่านั้น ซึ่งคาดว่าน่าจะเป็นผลมาจากจำนวนปีย้อนหลังที่น้อยเกินไปด้วย อัตราการเติบโตอาจไม่ใช่แนวเส้นตรง จึงควรจะเพิ่มจำนวนปีย้อนหลัง หรือแปลงค่าปีเป็นค่าอื่นๆ ก่อนจะนำมาใช้วิเคราะห์

- การนับวันหยุดนักขัตฤกษ์ (Holidays) เป็นการนับโดยคร่าวๆเท่านั้น ไม่ได้นับตามจำนวนวันหยุดนักขัตฤกษ์จริงในเดือนและปีนั้น การปรับแก้จำนวนวันหยุดอาจทำให้แม่นยำขึ้น หรืออาจจะมีวันประเภทอื่นๆเข้ามารวมด้วย เช่น ช่วงวันที่มีเทศกาลประเพณีในท้องถิ่น แต่ไม่ใช่วันหยุด หรือช่วงเทศกาลหยุดยาวติดต่อกันหลายวัน ที่อาจมีวันหยุดก่อนหรือหลังวันหยุดจริง หรือวันหยุดเพิ่มเติมตามประกาศของรัฐบาล เป็นต้น

- นอกจากนี้ อาจมีปัจจัยภายนอกที่คาดไม่ถึง แต่ส่งผลต่อค่าเป้าหมาย เช่น เศรษฐกิจ นโยบายขององค์กร หรือเหตุการณ์โดยรอบ อาจเป็นเพราะผู้จัดทำรู้รายละเอียดเกี่ยวกับชุดข้อมูลดังกล่าวไม่มากพอ ทำให้อาจยังขาดตัวแปรสำคัญที่จะทำนาย หรือไม่มีข้อมูลที่เกี่ยวข้องกับตัวแปรดังกล่าว หากศึกษาหัวข้อนี้ต่อควรจะคำนึงถึงปัจจัยอื่นๆให้มากขึ้น

# ภาคผนวก

# ภาคผนวก ก

# การติดตั้ง Java, Scala และ Apache Spark

คู่มือการติดตั้ง Java, Scala และ Spark ที่เขียนในรายงานเล่มนี้ จะใช้ในกรณีที่ติดตั้งบนระบบปฏิบัติการ Linux ซึ่งในที่นี้คือ Ubuntu 16.04 LTS

**การติดตั้ง Java**

การใช้งานภาษา Scala จำเป็นจะต้องติดตั้ง Java ด้วย โดยมีขั้นตอนดังนี้

1. เปิด terminal
2. ตรวจสอบ version ของ Java ได้ด้วยคำสั่ง

```
java -version
```

หากขึ้นผลลัพธ์เป็นเวอร์ชันแล้ว ให้ข้ามไปติดตั้ง Scala ได้เลย แต่หากยังไม่มีให้ทำขั้นตอนต่อไป

3. Update package index โดยใช้คำสั่ง

```
sudo apt-get update
```

ซึ่งจะต้องใส่ password ลงไปด้วย

4. ติดตั้ง JDK (Java Development Kit) ด้วยคำสั่ง

```
sudo apt-get install default-jdk
```

ซึ่งจะมีคำถาม Yes/No ให้ตอบ Y

**การติดตั้ง Scala**

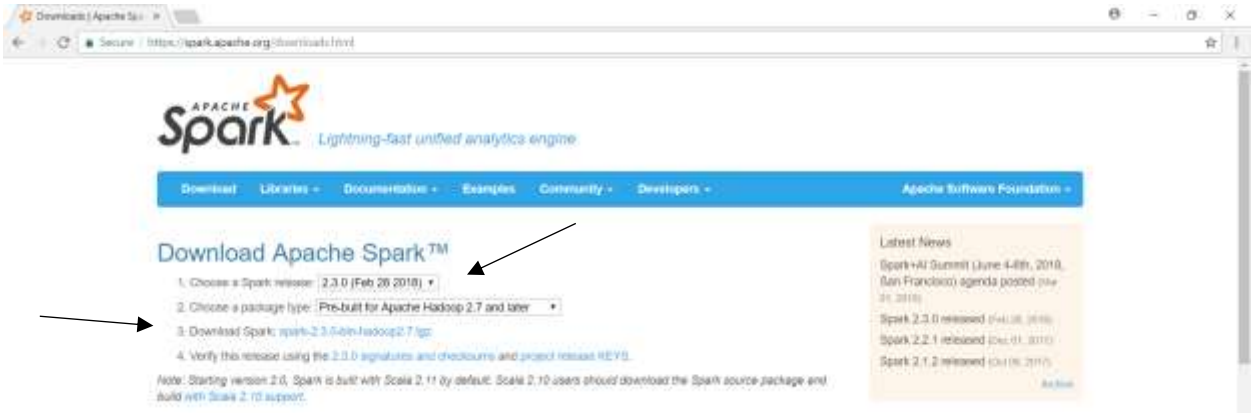
1. ใช้คำสั่ง

```
sudo apt-get install scala
```

2. สามารถตรวจสอบการติดตั้งโดยพิมพ์คำว่า scala ลงไป ซึ่งจะขึ้นว่าเปิด Scala ไว้ หากต้องการออกให้ใช้คำสั่ง :q

## การติดตั้ง Apache Spark

1. เข้าเว็บไซต์ https://spark.apache.org/ แล้วคลิกที่เมนู Download
2. เลือกเวอร์ชันที่ต้องการแล้วกดที่ลิงค์ถัดจากคำว่า “Download Spark: ”




3. จะมีหน้าจอแสดงลิงค์ดาวน์โหลดปรากฏขึ้น ให้เลือกลิงค์ที่แนะนำ ไฟล์ที่ดาวน์โหลดจะเป็นไฟล์ .tgz เมื่อดาวน์โหลดให้จำไว้ด้วยว่าโหลดลงที่ไหน


4. ติดตั้ง git ด้วยคำสั่ง

```
sudo apt-get install git
```

5. เปลี่ยน directory ให้เข้าไปในโฟลเดอร์ที่ดาวน์โหลดไฟล์ .tgz ไว้ แล้วแตกไฟล์โดยใช้คำสั่ง

```
tar xvf spark-2.0.2-bin-hadoop2.7.tgz
```

6. เมื่อแตกไฟล์เสร็จแล้ว สามารเปิด Spark Shell ได้โดยใช้คำสั่ง

```
cd spark-2.0.2-bin-hadoop2.7.tgz
cd bin
./spark-shell
```

7. หน้าจอจะปรากฏข้อมูลขึ้นว่ากำลัง run Spark อยู่ เป็นอันเสร็จสิ้น

# ภาคผนวก ข

# ตารางข้อมูลปัญหาจริง

**ตารางข้อมูลปี 2014 - 2015**

| Year | Month | A | B | C | Target |
|---|---|---|---|---|---|
| 2014 | 1 | 477 | 1618 | 0 | 5170.181447 |
| 2014 | 2 | 411 | 1479 | 0 | 4839.830647 |
| 2014 | 3 | 360 | 1515 | 0 | 4258.417334 |
| 2014 | 4 | 365 | 1350 | 0 | 3953.239897 |
| 2014 | 5 | 308 | 1365 | 0 | 3751.036583 |
| 2014 | 6 | 286 | 1274 | 0 | 3153.190983 |
| 2014 | 7 | 318 | 1296 | 0 | 3700.398261 |
| 2014 | 8 | 317 | 1425 | 0 | 4258.156889 |
| 2014 | 9 | 294 | 1398 | 0 | 3909.613679 |
| 2014 | 10 | 315 | 1376 | 0 | 3910.794965 |
| 2014 | 11 | 366 | 1441 | 0 | 4461.77512 |
| 2014 | 12 | 402 | 1537 | 0 | 4951.499809 |
| 2015 | 1 | 451 | 1595 | -2.34 | 5255.058833 |
| 2015 | 2 | 450 | 1485 | 2.38 | 4848.23188 |
| 2015 | 3 | 389 | 1555 | 3.68 | 4357.655835 |
| 2015 | 4 | 377 | 1408 | 4.08 | 4236.340641 |
| 2015 | 5 | 293 | 1462 | 4.9 | 3503.015867 |
| 2015 | 6 | 309 | 1340 | 5.71 | 3098.249546 |
| 2015 | 7 | 347 | 1404 | 8.49 | 3448.540932 |
| 2015 | 8 | 376 | 1597 | 13.26 | 3928.329692 |
| 2015 | 9 | 287 | 1499 | 5.56 | 3560.465057 |
| 2015 | 10 | 348 | 1544 | 11.89 | 3848.361062 |
| 2015 | 11 | 377 | 1606 | 9.74 | 4041.074781 |
| 2015 | 12 | 433 | 1703 | 10.16 | 4975.681475 |

ตารางที่ 4 : ข้อมูลปี 2014 - 2015

**ตารางข้อมูลปี 2016**

| Year | Month | A | B | C | Target |
|---|---|---|---|---|---|
| 2016 | 1 | 503 | 1782 | 11.68 | 6086.950549 |
| 2016 | 2 | 430 | 1564 | 3.05 | 5088.537844 |
| 2016 | 3 | 404 | 1764 | 11.52 | 5188.059433 |
| 2016 | 4 | 362 | 1621 | 11.09 | 4358.717488 |
| 2016 | 5 | 355 | 1791 | 22.28 | 4580.202723 |
| 2016 | 6 | 286 | 1569 | 12.49 | 4023.341888 |
| 2016 | 7 | 357 | 1528 | 7.65 | 4094.383801 |
| 2016 | 8 | 384 | 1612 | 1.17 | 4405.725665 |
| 2016 | 9 | 353 | 1511 | 4.37 | 4359.002172 |
| 2016 | 10 | 488.4 | 1821.4 | 22.08 | 4738.564216 |
| 2016 | 11 | 518.85 | 1889.6 | 21.45 | 4980.429535 |
| 2016 | 12 | 577.65 | 1996.3 | 20.5 | 6037.966206 |

ตารางที่ 5 : ข้อมูลปี 2016

# แหล่งที่มา

[1] *Linear Regression.* On-line. Available from Internet, https://en.wikipedia.org/wiki/Linear_regression, accessed 17 May 2018.

[2] *How should I delete rows from a DataFrame in Python-Pandas?* On-line. Available from Internet, https://www.quora.com/How-should-I-delete-rows-from-a-DataFrame-in-Python-Pandas, accessed 17 May 2018.

[3] *Norm (mathematics).* On-line. Available from Internet, https://en.wikipedia.org/wiki/Norm_(mathematics), accessed 17 May 2018.

[4] *The Area Under an ROC Curve*. On-line. Available from Internet, http://gim.unmc.edu/dxtests/roc3.htm), accessed 17 May 2018.

[5] *ค่าเฉลี่ยกำลังสอง*. ออนไลน์. สืบค้นจากอินเทอร์เน็ต, https://th.wikipedia.org/wiki/ค่าเฉลี่ยกำลังสอง, สืบค้นเมื่อ 17 พฤษภาคม 2561.

[6] http://vishnuviswanath.com/spark_lr.html

[7] https://github.com/soniclavier/bigdata-notebook/blob/update-spark-version/spark/src/main/scala/com/vishnu/spark/kaggle/titanic/KaggleTitanic.scala

[8] https://www.kaggle.com/c/titanic/data