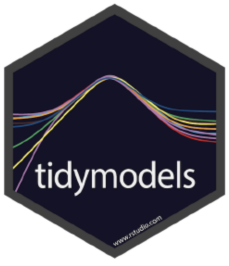
Machine Learning with tidymodels

สิวะโชติ ศรีสุทธิยากร

2023-01-06

Contents



อ้างอิง

- https://www.tmwr.org/index.html

การเรียนรู้ของเครื่อง (ML) เป็นศาสตร์ย่อยแขนงหนึ่งภายใต้ศาสตร์ทางด้านสถิติและวิทยาการข้อมูล ซึ่งเกี่ยวข้องกับการใช้**อัลกอริทึม (algorithms)** ในการเรียนรู้/ค้นหาความรู้จากข้อมูล แล้วนำความรู้ที่ได้มาใช้งานตั้งแต่การบรรยายสภาพของข้อมูล (descriptive) การวินิจฉัย (diagnostic) เพื่อหาสาเหตุหรือปัจจัยที่ก่อให้เกิดผลลัพธ์ที่สนใจ การทำนาย (predictive) เพื่อสร้างโมเดลที่เรียนรู้ความสัมพันธ์ในข้อมูลเพื่อทำนายผลลัพธ์ของตัวแปรที่สนใจ ผลลัพธ์ที่ได้จากการทำนายนี้สามารถนำมาโมเดลเพื่อช่วยวางแผน/ตัดสินใจ (prescriptive) ดำเนินการเพื่อนำไปสู่ผลลัพธ์ที่คาดหวัง

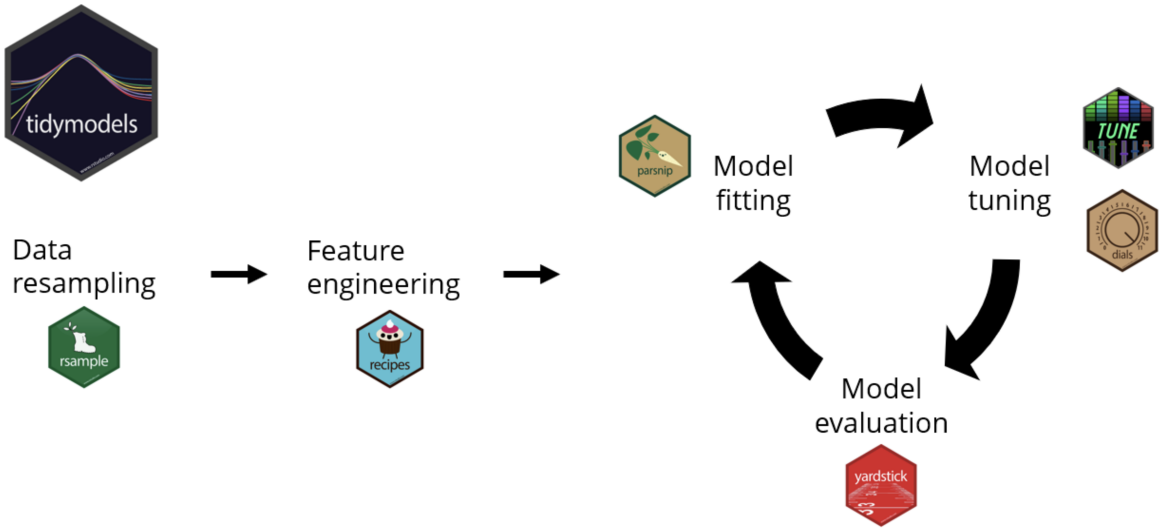
ตัวอย่างการประยุกต์ใช้ ML ทางการศึกษา

- https://iopscience.iop.org/article/10.1088/1757-899X/1031/1/012061/pdf
- https://peerj.com/articles/cs-986.pdf
- https://educationaltechnologyjournal.springeropen.com/articles/10.1186/s41239-020-00186-2
- https://bera-journals.onlinelibrary.wiley.com/doi/full/10.1002/rev3.3310
- https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9015108/

บทเรียนนี้จะกล่าวถึงการใช้โปรแกรม R เพื่อพัฒนาโมเดลการเรียนรู้ของเครื่องในกลุ่มของ supervised learning model วัตถุประสงค์หลักของการนำโมเดลในกลุ่มนี้ไปใช้คือการทำนายค่าสังเกตของตัวแปรตามที่ไม่ทราบค่า ด้วยข้อมูลที่ทราบค่าของตัวแปรอิสระ อย่างไรก็ตามก่อนที่จะมีโมเดลไปใช้ทำนายค่าสังเกตของตัวแปรตามได้นั้น ต้องมีการพัฒนาโมเดลทำนายดังกล่าวก่อน ขั้นตอนในการพัฒนาโมเดลทำนายอย่างคร่าว ๆ คือ ผู้วิเคราะห์จะต้องทำการเก็บรวบรวมข้อมูลของตัวแปรตามกับตัวแปรอิสระจากกลุ่มเป้าหมายหรือประชากรที่สนใจ จากนั้นนำข้อมูลดังกล่าวมาสอนให้โมเดลหรืออัลกอริทึมในกลุ่มของ supervised learning เช่น linear regression, logistic regression, K-NN, decision tree, random forest หรือ neural network ได้เรียนรู้รูปแบบความสัมพันธ์ระหว่างตัวแปรตามกับตัวแปรอิสระดังกล่าวก่อน จากนั้นจึงนำโมเดลทำนายที่ผ่านการพัฒนาแล้วไปใช้งาน

Tidymodels

tidymodel เป็น framework ที่ถูกพัฒนาขึ้นเพื่อสนับสนุนการสร้าง machine learning model ในโปรแกรม R ภายใต้ framework นี้ประกอบด้วย package หลายตัว โดยที่แต่ละตัวที่หน้าที่สำหรับทำงานในส่วนต่าง ๆ ภายใต้กระบวนการพัฒนา โมเดล machine learning แผนภาพด้านล่างแสดง framwork ของ tidymodels ดังกล่าว ซึ่งจะเห็นว่าครอบคลุมกระบวนการพัฒนา machine learning model ทั้งหมด



- **package-rsample** ใช้ในงาน resampling ข้อมูล เช่นการสร้าง training/validation/test dataset การสร้าง cross-validation dataset หรือการสร้าง bootstrape dataset
- **package-recipes** ใช้แปลง/แก้ปัญหาที่เกิดขึ้นในข้อมูลของตัวแปรที่ใช้ในการพัฒนาโมเดล ขั้นตอนนี้เรียกว่า feature engineering
- **package-parsnip** ใช้ fit machine learning กับข้อมูล
- **package-Tune** และ **package-dials** มีฟังก์ชันที่อำนวยความสะดวกในการ fine tune hyperparameter ของโมเดลเพื่อเพิ่มประสิทธิภาพการทำนายของโมเดลให้สูงที่สุด
- **package-yardstick** มีฟังก์ชันของ metric ที่ใช้ประเมินประสิทธิภาพของโมเดลทำนาย

tidymodels ถูกพัฒนาขึ้นโดยได้รับการออกแบบให้สามารถทำซ้ำกระบวนการพัฒนาโมเดลได้ง่าย โดยใช้ไวยกรณ์ของภาษาในลักษณะเดียวกัน และถูกออกแบบโดยเน้นใช้กับ supervised learning เป็นหลัก

ผู้ใช้งานไม่จำเป็นต้องติดตั้งทุก package ในข้างต้นด้วยตนเอง แต่ติดตั้งเพียง package-tidymodels ก็สามารถใช้งานทุก package ภายใต้ framework ดังกล่าวได้แล้ว โดยการพิมพ์คำสั่งต่อไปนี้

```
install.packages("tidymodels") # ดาวน์โหลดและติดตั้ง tidymodels
library(tidymodels) # เรียกใช้ tidymodels
```

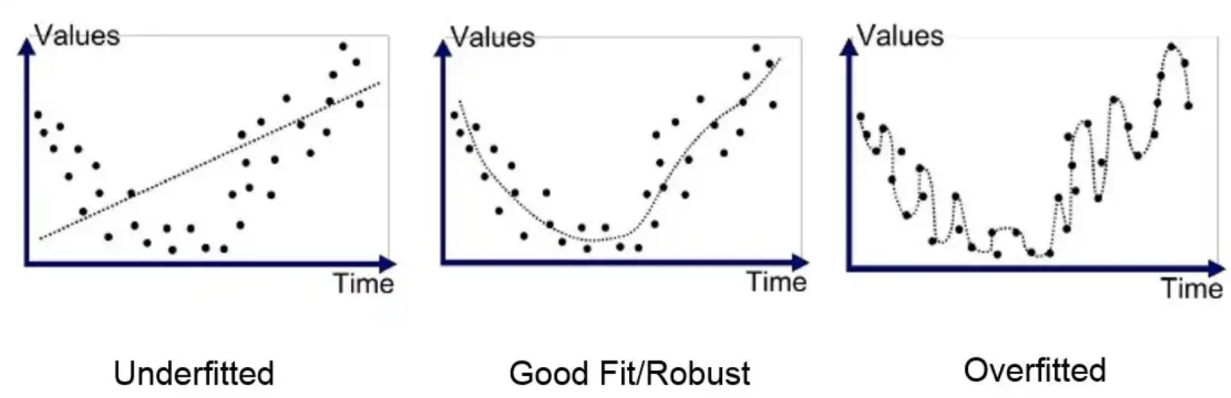
ประเภทของ supervised learning

supervised learning อาจจำแนกได้เป็นสองประเภท ได้แก่ regression และ classification ความแตกต่างของ supervised learning ทั้งสองอยู่ที่ตัวแปรตามที่ต้องการทำนายค่าเป็นตัวแปรแบบเชิงปริมาณ หรือแบบจัดประเภท โดย regression model จะใช้กับตัวแปรตามแบบเชิงปริมาณ ส่วน classification model จะใช้กับตัวแปรตามแบบจัดประเภท

Bias and Variance

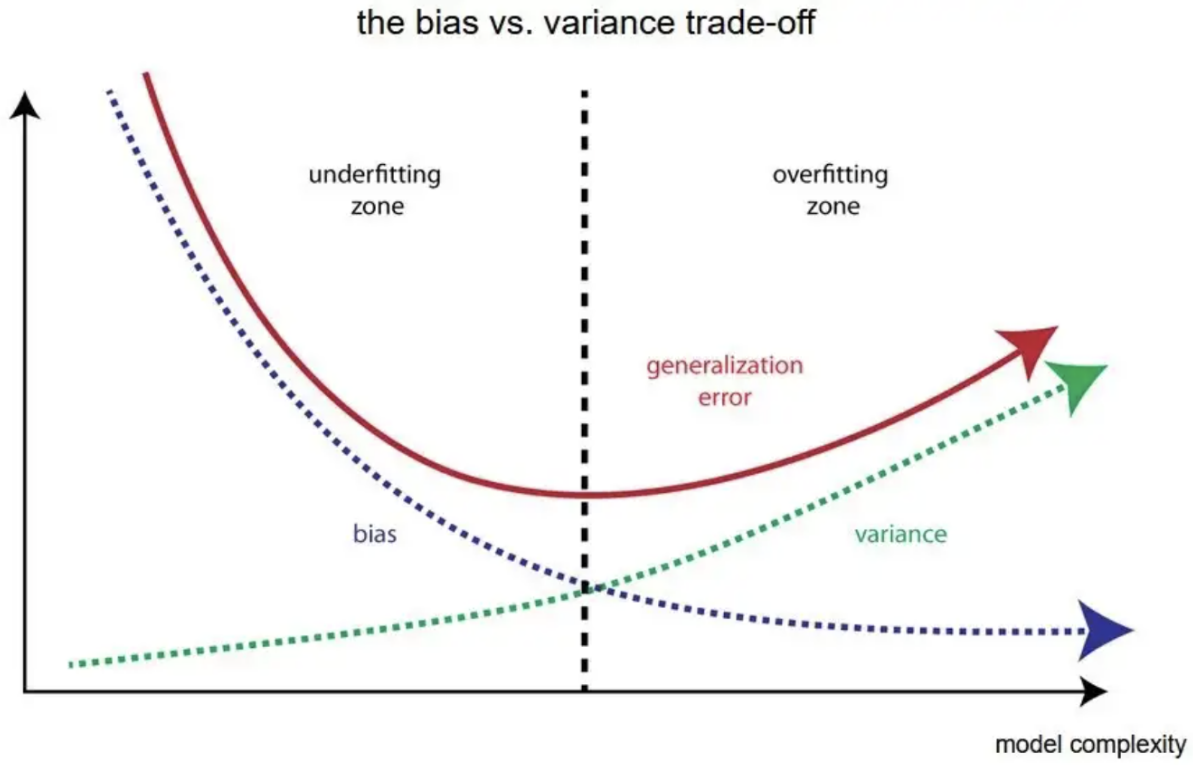
การวัดประสิทธิภาพของโมเดลทำนายอาจพิจารณาจากความคลาดเคลื่อนสองตัวได้แก่ ความลำเอียง (biased) และความแปรปรวน (variance) ความลำเอียงเป็นความคลาดเคลื่อนระหว่างค่าทำนายที่ได้จากโมเดลกับค่าสังเกตจริงของชุดข้อมูลฝึกหัดที่ใช้สอนโมเดล ส่วนความแปรปรวนเป็นความคลาดเคลื่อนระหว่างค่าทำนายของโมเดลกับค่าสังเกตจริงในชุดข้อมูลทดสอบ

ในกระบวนการพัฒนาโมเดลทำนาย ผู้วิเคราะห์จะต้องมีชุดข้อมูลตัวอย่าง ซึ่งต่อไปจะเรียกว่าชุดข้อมูลฝึกหัด (training data) ที่ภายในชุดข้อมูลประกอบด้วยข้อมูลตัวแปรตามที่เป็นเป้าหมายของการทำนาย และตัวแปรอิสระที่จะใช้เป็นข้อมูลสำหรับทำนายตัวแปรตาม โดยจะใช้ชุดข้อมูลฝึกหัดดังกล่าวเป็นแบบฝึกหรือตัวอย่างให้อัลกอริทึมต่าง ๆ เรียนรู้รูปแบบความสัมพันธ์ระหว่างตัวแปรตามกับตัวแปรอิสระภายในชุดข้อมูล อัลกริทึมที่เรียนรู้ได้ดีจะให้โมเดลทำนายที่มีความสอดคล้องกับความสัมพันธ์ในชุดข้อมูลฝึกหัดสูง (หรือมีความลำเอียงต่ำ) เช่นดังรูปกลางและรูปขวามือด้านล่าง ในทางกลับกันอัลกอริทึมที่เรียนรู้ได้ไม่ดีจะให้โมเดลทำนายที่มีความสอดคล้องกับข้อมูลต่ำ (หรือมีความลำเอียงสูง) เช่นดังรูปซ้ายมือด้านล่าง ในเชิงเทคนิคเรียกโมเดลที่มีความลำเอียงสูงนี้ว่า underfitting model



อย่างไรก็ตามโมเดลที่มีความลำเอียงต่ำอาจไม่ใช่โมเดลที่มีประสิทธิภาพในการทำนายเสมอไป จากรูปด้านบนจะเห็นว่า โมเดลในรูปทางขวามือสามารถเรียนรู้ความสัมพันธ์ระหว่าง Values กับ Time ได้อย่างสมบูรณ์ กล่าวคือภายในชุดข้อมูลฝึกหัดที่นำมาให้โมเดลนี้เรียนรู้ ไม่มีหน่วยข้อมูลใดเลยที่โมเดลทำนายคลาดเคลื่อน ซึ่งเหมือนว่าเราพัฒนาโมเดลทำนายได้อย่างสมบูรณ์แบบมาก ๆ แต่เมื่อคิดในมุมของการนำโมเดลไปใช้ในกลุ่มเป้าหมาย จะพบว่าเมื่อนำโมเดลไปทำนายหน่วยข้อมูลอื่นที่อยู่นอกเหนือชุดข้อมูลฝึกหัด มีโอกาสที่โมเดลจะทำนายหน่วยข้อมูลดังกล่าวคลาดเคลื่อนในระดับที่สูงมาก (หรือมีความแปรปรวนสูง) ทั้งนี้เป็นเพราะโมเดลทำนายมีความยืดหยุ่นต่ำมากกับหน่วยข้อมูลอื่น ๆ ที่อยู่นอกเหนือจากชุดข้อมูลฝึกหัด กล่าวสั้น ๆ คือ โมเดลประเภทนี้มีความเป็นนัยทั่วไปอยู่ในระดับต่ำ ในเชิงเทคนิคเรียกโมเดลประเภทนี้ว่า overfitting model

ดังนั้นในกระบวนการพัฒนาโมเดลทำนาย จึงจำเป็นที่จะต้องมีการวิเคราะห์ความคลาดเคลื่อนทั้งในส่วนของความลำเอียง และความแปรปรวน เพื่อให้ได้ผลลัพธ์โมเดลทำนายที่มีประสิทธิภาพสูงที่สุด ทั้งนี้ในเชิงอุดมคติ ผู้วิเคราะห์ต้องการให้ความคลาดเคลื่อนทั้งส่วนที่เป็นความลำเอียง และความแปรปรวนนั้น มีความต่ำที่สุดเท่าที่จะสามารถต่ำได้ อย่างไรก็ตามความคลาดเคลื่อนทั้งสองนั้นไม่สามารถทำได้ต่ำที่สุดพร้อมกันได้ รูปด้านล่างแสดงความสัมพันธ์ระหว่างความลำเอียง และความแปรปรวน ซึ่งจะเห็นว่ามีการแปรผกผันซึ่งกันและกัน วัตถุประสงค์ของการพัฒนาโมเดลจึงเป็นการหาจุดที่ดีที่สุดที่ทำให้ความคลาดเคลื่อนทั้งสองอยู่ในจุดที่ต่ำที่สุดเท่าที่จะเป็นไปได้



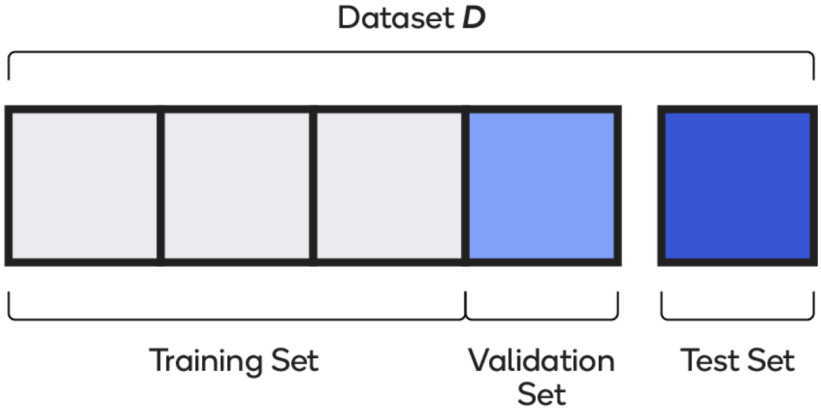
Data resampling

จาก concept ข้างต้น ทำให้พอสรุปได้ว่า ความลำเอียงเป็นความคลาดเคลื่อนในการทำนายหน่วยข้อมูลภายในชุดข้อมูลฝึกหัดของโมเดล ส่วนความแปรปรวนเป็นความคลาดเคลื่อนในการทำนายหน่วยข้อมูลที่อยู่นอกเหนือจากชุดข้อมูลฝึกหัดของโมเดล การที่จะวิเคราะห์ความคลาดเคลื่อนทั้งสองให้ได้นั้น ผู้วิเคราะห์จึงจำเป็นจะต้องมีชุดข้อมูลอย่างน้อยสองชุด ได้แก่ ชุดข้อมูลฝึกหัด (training dataset) ที่จะใช้เป็นตัวอย่างหรือข้อมูลให้โมเดลเรียนรู้ และชุดข้อมูลทดสอบ (test dataset) ที่จะใช้เป็นชุดข้อมูลสำหรับตรวจสอบประสิทธิภาพในการทำนายเท่านั้น ไม่ถูกนำมาใช้ในขั้นของการเรียนรู้ของโมเดล ดังนั้นความคลาดเคลื่อนที่เกิดขึ้นจากการทำนายในชุดข้อมูลทดสอบจึงสามารถใช้สะท้อนความแปรปรวนของโมเดลตาม concept ในหัวข้อก่อนหน้านี้ได้

ในทางปฏิบัติผู้วิเคราะห์มีข้อมูลต้นฉบับเพียงชุดเดียว แนวทางปฏิบัติโดยมากผู้วิเคราะหจึงจะแบ่งข้อมูลต้นฉบับดังกล่าวออกเป็นสองส่วน ได้แก่ ชุดข้อมูลฝึกหัด และชุดข้อมูลทดสอบ โดยจะใช้การสุ่มตัวอย่าง (random sampling) เป็นเครื่องมือในการแบ่งส่วนข้อมูลดังกล่าว

อย่างไรก็ตามอัลกอริทึมการเรียนรู้ของเครื่องหลายตัวมีส่วนประกอบหนึ่งที่เรียกว่า hyperparameter ที่ใช้ควบคุมลักษณะการเรียนรู้ของแต่ละอัลกอริทึม hyperparameter นี้มีความแตกต่างไปจาก parameter ทั่วไปคือไม่สามารถประมาณค่าได้จากข้อมูลด้วยวิธีการทางสถิติ แต่จะต้องใช้การปรับแต่งค่าโดยผู้วิเคราะห์เอง เพื่อให้ค่าของ hyperparameter ที่ปรับแต่งมีความเป็นนัยทั่วไปด้วย จึงจะมีการแบ่งส่วนของ training dataset ออกเป็นสองส่วน ได้แก่ ส่วนที่เป็น training dataset และส่วนของ validation set โดยการประมาณค่า parameter หรือ train โมเดลทำนายจะใช้เฉพาะส่วน training dataset เท่านั้น และ

จะวัดประสิทธิภาพจากชุด validation set ก่อนเพื่อใช้เป็นแนวทางในการกำหนด/ปรับแต่งค่า hyperparameter ภายในโมเดล และเมื่อพัฒนาโมเดลในขั้นตอนนี้เสร็จแล้วจึงนำโมเดลทำนายไปตรวจสอบความเป็นนัยทั่วไปด้วยชุดข้อมูลทดสอบอีกครั้งหนึ่ง



ในเชิงปฏิบัติการแบ่งส่วนข้อมูลออกเป็น training และ test data ไม่ได้มีอัตราส่วนที่แน่นอนทั้งนี้ขึ้นอยู่กับว่าชุดข้อมูลต้นฉบับที่นำมาวิเคราะห์มีขนาดใหญ่/เล็กมากน้อยเพียงใด ideal คือเราต้องการชุดข้อมูล training data ที่ใหญ่เพียงพอจะทำให้อัลกอริทึมสามารถเรียนรู้ความสัมพันธ์ระหว่างข้อมูลได้อย่างครบถ้วน ครอบคลุม และต้องการ test data ที่มีความเป็นตัวแทนกลุ่มเป้าหมาย เพื่อมั่นใจได้ว่าผลการประมาณค่าความคลาดเคลื่อนจะสะท้อนประสิทธิภาพของโมเดลภายในกลุ่มเป้าหมายได้อย่างแม่นยำ อัตราส่วนที่มักใช้กันเช่น 80 : 20, 75 : 25, 60 : 40 หรือ 50 : 50 เป็นต้น การแบ่งส่วนข้อมูลดังกล่าวอาจทำได้สองวิธีการ วิธีการแรกคือการแบ่งด้วยการสุ่มอย่างง่าย (simple random sampling) และวิธีการที่สองคือการแบ่งด้วนการสุ่มแบบชั้นภูมิ (stratified random sampling)

ชุดข้อมูล `mpg`

ชุดข้อมูลที่ใช้เป็นตัวอย่างจะใช้ dataset `mpg` ของ R ซึ่งถูกติดตั้งมาพร้อมกับการติดตั้งโปรแกรม R อยู่แล้ว ผู้วิเคราะห์สามารถเรียกดูข้อมูลภายใต้ชุดข้อมูล `mpg` ได้โดยพิมพ์คำสั่งต่อไปนี้

```
head(mpg)
```

```
## # A tibble: 6 x 11
##   manufacturer model displ  year   cyl trans      drv     cty   hwy fl    class
##   <chr>        <chr> <dbl> <int> <int> <chr>      <chr> <int> <int> <chr> <chr>
## 1 audi         a4      1.8  1999     4 auto(l5)   f        18    29 p     compact
## 2 audi         a4      1.8  1999     4 manual(m5) f        21    29 p     compact
## 3 audi         a4      2    2008     4 manual(m6) f        20    31 p     compact
## 4 audi         a4      2    2008     4 auto(av)   f        21    30 p     compact
## 5 audi         a4      2.8  1999     6 auto(l5)   f        16    26 p     compact
## 6 audi         a4      2.8  1999     6 manual(m5) f        18    26 p     compact
```

```
glimpse(mpg)
```

```
## Rows: 234
## Columns: 11
## $ manufacturer <chr> "audi", "audi", "audi", "audi", "audi", "audi", "audi", "audi", "audi", "~
## $ model        <chr> "a4", "a4", "a4", "a4", "a4", "a4", "a4", "a4 quattro", "a4 quattro", "a4~
## $ displ        <dbl> 1.8, 1.8, 2.0, 2.0, 2.8, 2.8, 3.1, 1.8, 1.8, 2.0, 2.0, 2.8, 2.8, 3.1, 3.1~
## $ year         <int> 1999, 1999, 2008, 2008, 1999, 1999, 2008, 1999, 1999, 2008, 2008, 1999, 1~
## $ cyl          <int> 4, 4, 4, 4, 6, 6, 6, 4, 4, 4, 4, 6, 6, 6, 6, 6, 6, 8, 8, 8, 8, 8, 8, 8, 8~
## $ trans        <chr> "auto(l5)", "manual(m5)", "manual(m6)", "auto(av)", "auto(l5)", "manual(m~
## $ drv          <chr> "f", "f", "f", "f", "f", "f", "f", "4", "4", "4", "4", "4", "4", "4", "4"~
## $ cty          <int> 18, 21, 20, 21, 16, 18, 18, 18, 16, 20, 19, 15, 17, 17, 15, 15, 17, 16, 1~
## $ hwy          <int> 29, 29, 31, 30, 26, 26, 27, 26, 25, 28, 27, 25, 25, 25, 25, 24, 25, 23, 2~
## $ fl           <chr> "p", "p", "p", "p", "p", "p", "p", "p", "p", "p", "p", "p", "p", "p", "p"~
## $ class        <chr> "compact", "compact", "compact", "compact", "compact", "compact", "compac~
```

จากผลการสำรวจข้างต้นจะเห็นว่าชุดข้อมูล `mpg` ประกอบด้วยข้อมูลเกี่ยวกับรถยนต์ โดยมีตัวแปร (หรือ features) ทั้งหมด 11 ตัว และมีหน่วยข้อมูล (หรือ instance) จำนวน 234 หน่วยข้อมูล ในตัวอย่างนี้กำหนดให้ตัวแปรตามที่ต้องการทำนายคือ `hwy` (highway fuel efficiency in miles per gallon) ซึ่งเป็นประสิทธิภาพในด้านการประหยัดพลังงานของรถยนต์แต่ละคัน

Simple random sampling

การแบ่งด้วย simple random sampling เป็นการแบ่งโดยสุ่มข้อมูลตามจำนวนที่กำหนดออกมาเป็นชุดข้อมูล training dataset หรือ test dataset โดยการสุ่มดังกล่าวมีข้อสมมุติว่าหน่วยข้อมูลทุกหน่อยในชุดข้อมูลต้นฉบับมีโอกาสที่จะถูกสุ่มขึ้นมาเท่ากันทั้งหมด การแบ่งข้อมูลด้วยวิธีการนี้ใน R สามารถทำได้หลายวิธี แต่ในบทความนี้จะใช้วิธีที่อยู่ภายใต้ framework ของ tidymodels โดยใช้ฟังก์ชัน `initial_split()`

ฟังก์ชันดังกล่าวใช้สำหรับสร้างกรอบการแบ่งข้อมูลด้วยการสุ่มอย่างง่าย ฟังก์ชันดังกล่าวมีอาร์กิวเมนท์ที่สำคัญได้แก่ `data` สำหรับระบุชุดข้อมูลแบบ data.frame ที่ต้องการแบ่ง และ `prop` ใช้ระบุสัดส่วนของ training dataset (เช่น 0.75, 0.8 เป็นต้น) เมื่อสร้างกรอบการแบ่งข้อมูลแล้วส่งผ่านผลลัพธ์ของ `initial_split()` ไปยังฟังก์ชัน `training()` และ `testing()` เพื่อสร้างชุดข้อมูล training และ test dataset ต่อไปนี้ ตัวอย่างด้านข้างแสดงการเขียนคำสั่งตามขั้นตอนการดำเนินงานดังที่กล่าวมา

```
mpg_split1 <- initial_split(data = mpg, prop = 0.75)
mpg_split1
```

```
## <Training/Testing/Total>
## <175/59/234>
```

ผลลัพธ์ในข้างต้นแสดงให้เห็นว่าฟังก์ชัน `initial_split()` สร้างกรอบการแบ่งข้อมูล โดยกำหนดให้ training dataset มี

จำนวน 175 หน่วยข้อมูล และ test dataset มีจำนวน 59 หน่วยข้อมูล

```
train_srs <- mpg_split1 %>% training()
test_srs <- mpg_split1 %>% testing()
```

Stratified random sampling

วัตถุประสงค์หลักของการพัฒนา machine learning model คือการหาโมเดลที่สามารถทำนายตัวแปรตามได้อย่างมีประสิทธิภาพ ภายในชุดข้อมูล training และ test dataset จึงควรมีหน่วยข้อมูลที่มีค่าของตัวแปรตามครอบคลุมค่าที่เป็นไปได้ของตัวแปรตามทั้งหมด วิธีการสุ่มแบบชั้นภูมิเป็นวิธีการที่ถูกนำมาใช้เพื่อช่วยรับประกันว่าชุดข้อมูล training และ test จะมีหน่วยข้อมูลที่ครอบคลุมค่าที่เป็นไปได้ของตัวแปรตามได้ครบหรือใกล้เคียงกันมากที่สุด

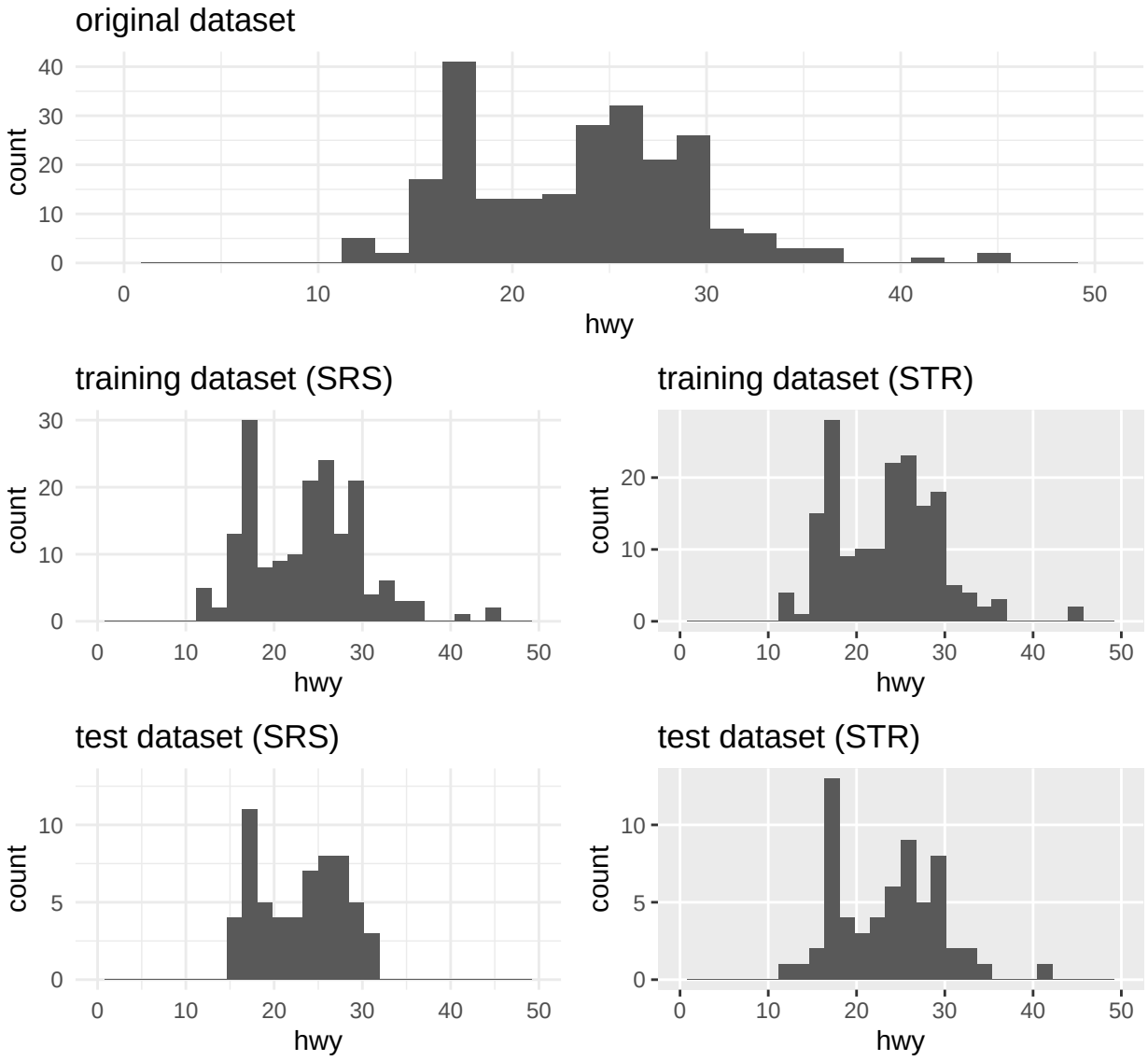
การแบ่งชุดข้อมูลด้วยการสุ่มแบบชั้นภูมิสามารถทำได้ด้วยฟังก์ชัน `initial_split()` เช่นเดียวกัน แต่จะต้องมีการระบุอาร์กิวเมนท์ `strata` เป็นตัวแปรเกณฑ์ที่จะใช้แบ่งชั้นภูมิซึ่งมักใช้เป็นตัวแปรตามของโมเดลด้วยเหตุผลดังที่กล่าวมาแล้ว นอกจากนี้ยังมีอาร์กิวเมนท์ `breaks` ที่ใช้ระบุจำนวนอันตรภาคชั้นที่จะใช้แบ่งชั้นภูมิของตัวแปรตามในกรณีที่ตัวแปรตามเป็นเชิงปริมาณ ค่าเริ่มต้นของอาร์กิวเมนท์นี้กำหนดให้ `breaks = 4`

ตัวอย่างต่อไปนี้แสดงการแบ่งชุดข้อมูล training และ test โดยใช้การสุ่มแบบชั้นภูมิ

```
mpg_split2 <- initial_split(data = mpg,
                            prop = 0.75,
                            strata = "hwy",
                            breaks = 8)

train_str <- training(mpg_split2)
test_str <- testing(mpg_split2)
```

รูปด้านล่างแสดงการเปรียบเทียบการแจกแจงของตัวแปรตามระหว่างชุดข้อมูลต้นฉบับ และชุดข้อมูล training และ test ที่แบ่งด้วยวิธีการสุ่มอย่างง่าย และสุ่มแบบชั้นภูมิ



Linear Regression with tidymodels

หัวข้อนี้จะแสดงการพัฒนาโมเดลทำนายด้วยอัลกอริทึม linear regression โดยใช้ Tidymodels ที่ได้กล่าวไปในข้างต้น การ train ให้อัลกอริทึม linear regression สามารถเรียนรู้ความสัมพันธ์ระหว่างตัวแปรตาม `hyw` กับตัวแปรทำนายต่าง ๆ ในชุดข้อมูลจะดำเนินงานด้วย package parsnip และการประเมินประสิทธิภาพในการทำนายของโมเดลที่พัฒนาขึ้นจะใช้ evaluation metric ที่อยู่ภายใต้ package-yardstick

โมเดลการวิเคราะห์การถดถอยเชิงเส้น (linear regression) เป็นโมเดลเชิงสถิติ (statistical model) ที่ใช้สำหรับทำนายแนวโน้มค่าสังเกตของตัวแปรตามที่ไม่ทราบค่าโดยอิงกับค่าสังเกตของตัวแปรอิสระที่ทราบค่า การเรียนรู้ของ linear regression จะพยายามสร้างสมการเส้นตรงที่ดีที่สุด (best linear equation) ที่สามารถใช้เป็นตัวแทนความสัมพันธ์ตามธรรมชาติระหว่างตัวแปรตามกับตัวแปรอิสระที่พบในชุดข้อมูล ในเชิงเทคนิคการหาสมการเส้นตรงดังกล่าวจะเป็นการแก้สมการหรือเฟ้นหาค่าของพารามิเตอร์ภายในสมการเส้นตรง ได้แก่ พารามิเตอร์จุดตัดแกน y และพารามิเตอร์ความชัน ที่ทำให้สมการเส้นตรงมีความคลาด

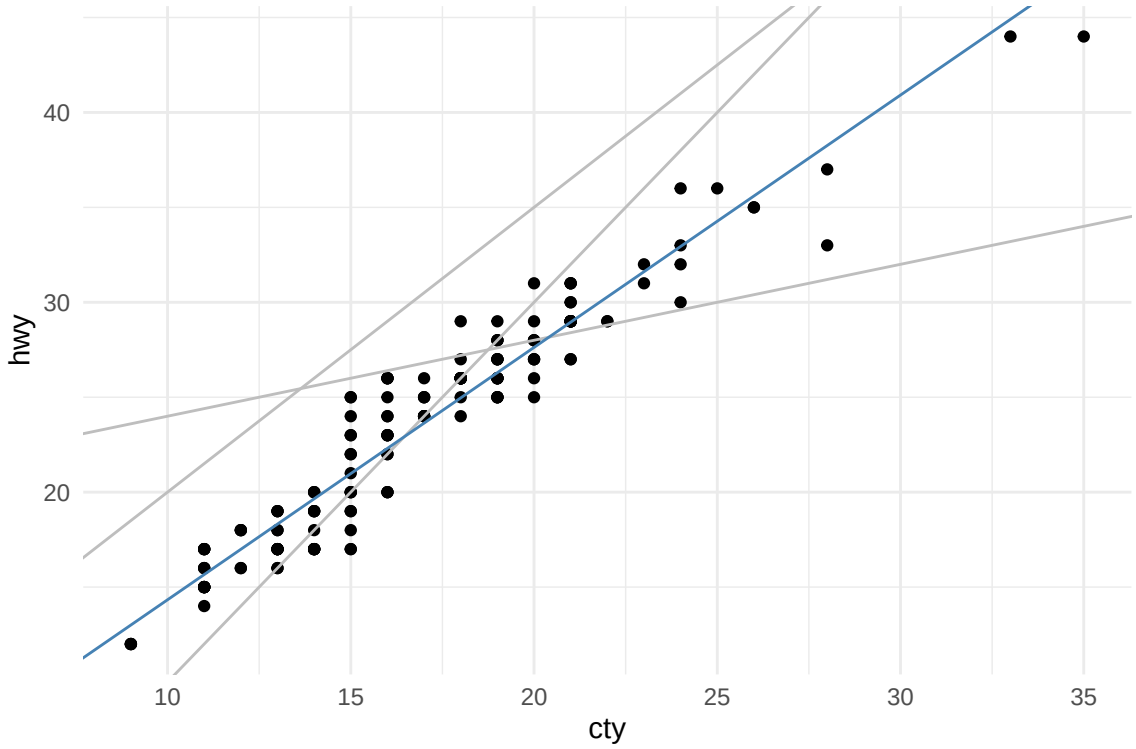
เคลื่อนในการทำนายต่ำที่สุด

สมการเส้นตรงดังกล่าวจะถูกใช้เป็นเครื่องมือสำหรับทำนายค่าของตัวแปรตามเมื่อกำหนดค่าสังเกตที่ทราบค่าของตัวแปรอิสระทั้งหมดภายในโมเดล ยกตัวอย่างเช่น จากชุดข้อมูล `mpg` สมมุติว่าผู้วิเคราะห์เลือกให้ตัวแปร `cty` (city fuel efficiency) เป็นตัวแปรอิสระเพียงตัวเดียวที่จะใช้ทำนาย `hwy` กรณีนี้สมการเส้นตรงที่จะใช้ทำนายสามารถเขียนในรูปมาตรฐานได้ดังนี้

$$hwy = \beta_0 + \beta_1 cty$$

โดยที่ β_0, β_1 คือพารามิเตอร์จุดตัดแกน และความชันของสมการ จากรูปด้านล่างจะเห็นว่า การกำหนดค่าพารามิเตอร์ดังกล่าวที่แตกต่างกันจะทำให้ได้ผลลัพธ์เป็นเส้นตรงที่มีจุดตัดแกนและความชันที่แตกต่างกัน ซึ่งมีโดยตรงผลต่อประสิทธิภาพในการทำนาย อัลกอริทึมจะหาค่าพารามิเตอร์ β_0, β_1 ที่ดีที่สุด ซึ่งจากรูปจะเห็นว่าเส้นตรงที่น้ำเงินคือเส้นตรงที่มีความสอดคล้องหรือ fit กับข้อมูลมากที่สุดเมื่อเปรียบเทียบกับเส้นตรงอื่น ๆ ภายใต้เส้นตรงทั้งหมดในรูป เส้นตรงสีน้ำเงินจึงเป็นเส้นตรงที่ดีที่สุด สมการเส้นตรงสีน้ำเงินสามารถเขียนได้ดังนี้

$$\hat{hwy} = 1.039 + 1.329 cty$$



อย่างไรก็ตามในกรณีทั่วไปเส้นตรงที่เป็นไปได้ของแต่ละชุดข้อมูลมีจำนวนนับไม่ถ้วน การหาเส้นตรงที่ดีที่สุดจึงเป็นการหาผ่านอัลกอริทึม โดยอัลกอริทึม linear regression จะหาค่าของค่าพารามิเตอร์ β_0, β_1 ที่ทำให้ฟังก์ชันผลรวมกำลังสอง (sum square errors: SSE) ที่มีต่ำที่สุด ภายใต้สถานการณ์ที่ต้องการใช้ `cty` เป็นตัวแปรเพื่อทำนาย `hyw` สามารถเขียนฟังก์ชัน SSE ได้ดังนี้

$$SSE = \sum_{i=1}^{n} (hwy - \hat{hwy})^2 = \sum_{i=1}^{n} (hwy - \beta_0 - \beta_1 cty)^2$$

รูปด้านล่างแสดงระนาบของฟังก์ชัน SSE ในแต่ละค่าที่เป็นไปได้ของพารามิเตอร์ β_0, β_1 ซึ่งจะเห็นว่าค่าพารามิเตอร์ที่เหมาะสมเท่านั้นที่จะทำให้ค่า SSE มีค่าต่ำที่สุดได้

การ fit linear regression ด้วย `parsnip`

คู่มือ package parsnip

- https://cran.r-project.org/web/packages/parsnip/parsnip.pdf
- https://cran.r-project.org/web/packages/parsnip/vignettes/parsnip.html

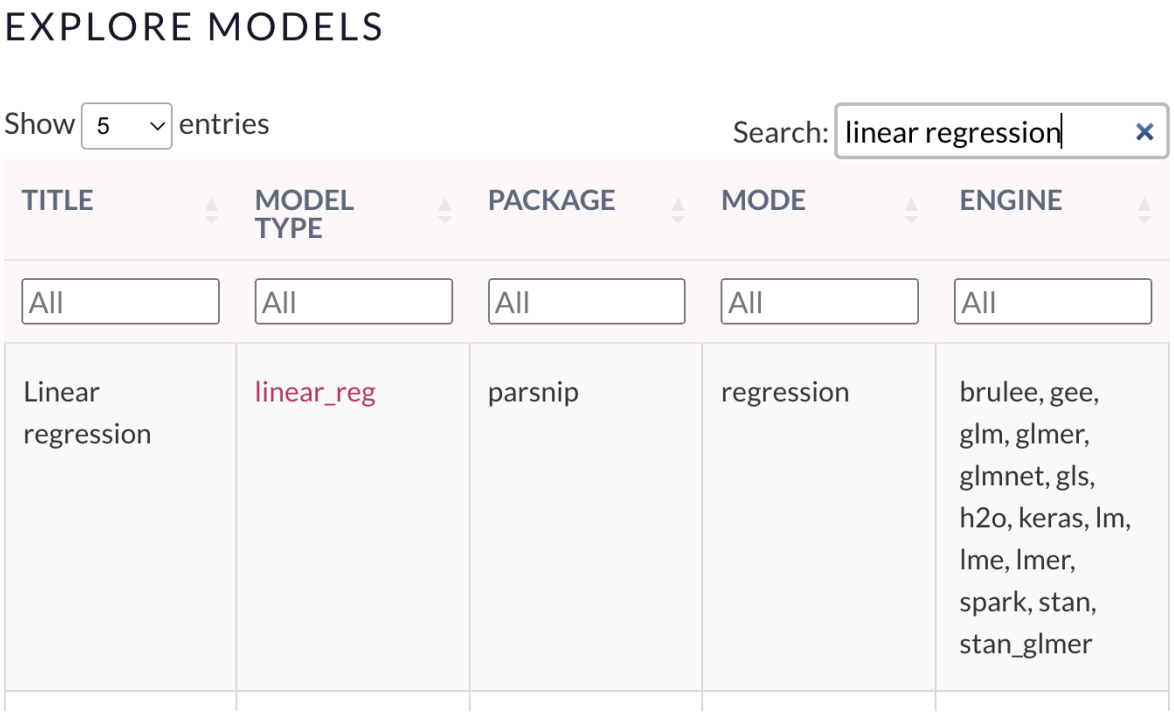
การ train อัลกอริทึมการเรียนรู้ของเครื่องใน R ที่ยุคเริ่มต้น ผู้วิเคราะห์ต้องทำงานร่วมกับ package หลายตัว ทั้งนี้เป็นเพราะแต่ละ package ใช้สำหรับ fit supervised learning ได้จำกัด ทั้งนี้ package ต่าง ๆ มักมีไวยกรณ์การเขียนคำสั่งที่แตกต่างกัน ทำให้ทำงานได้ยากพอสมควร package parsnip ถูกพัฒนาขึ้นเพื่อแก้ปัญหาดังกล่าว โดยทำหน้าที่เป็น interface ให้ผู้ใช้พิมพ์คำสั่งเพื่อใช้งาน package ของ R ต่าง ๆ ที่เกี่ยวข้องได้ ภายใต้ไวยกรณ์แบบเดียวกันทั้งหมด ซึ่งทำการทำงานง่ายขึ้นอย่างมาก ขั้นตอนการ fit model ด้วย parsnip จำแนกเป็นสองส่วน ส่วนแรกคือขั้นตอนการระบุโมเดล (model specification) และส่วนที่สองคือขั้นตอนการประมวลผล รายละเอียดมีดังนี้

1. การระบุโมเดลด้วย parsnip

การระบุโมเดลใน parsnip มีส่วนประกอบ 3 ส่วนที่จำเป็นได้แก่

1. **model type** ที่จะใช้ เช่น linear regression, K-NN, decision tree หรืออื่น ๆ
2. **engine** ที่จะใช้การประมวลผล
3. **mode** ว่าการทำนายเป็นแบบ regression หรือ classification

รายละเอียดว่าผู้วิเคราะห์สามารถกำหนด model type, engine และ mode แบบใดได้บ้างและต้องกำหนดอย่างไร สามารถศึกษาได้จาก https://www.tidymodels.org/find/parsnip/ รูปด้านล่างแสดงค้นหาสำหรับอัลกอริทึม linear regression



จากผลการค้นหาข้างต้นจะเห็นว่าการ fit linear regression ด้วย parsnip สามารถทำได้ด้วย model type คือ `linear_reg()` เมื่อพิจารณาในคอลัมน์ engine จะเห็นว่าการ fit linear regression มี engine จำนวนมากที่สามารถใช้เพื่อประมาณค่าพารามิเตอร์ของโมเดลได้ engine ดังกล่าวจริง ๆ แล้วคือ package ต่าง ๆ ของ R ที่ใช้ประมวลผล mode type ที่เลือกไว้ได้ ผู้อ่านจะเห็นว่า model type แบบ `linear_reg` มี engine ที่สามารถใช้ประมวลผลได้จำนวนมาก ซึ่งมีความเหมือนและความแตกต่างกัน เนื้อหาส่วนนี้มีความละเอียดและลึกมาก จึงขอไม่กล่าวถึงในที่นี้

ในคู่มือข้างต้นยังมีเครื่องมือให้ค้นหาการกำหนดอาร์กิวเมนท์ของฟังก์ชัน model type ในข้างต้น จากรูปด้านล่างจะเห็นรายละเอียดในการกำหนดอาร์กิวเมนท์ของฟังก์ชัน `linear_reg()` เมื่อกำหนด engine ในลักษณะต่าง ๆ

Show 5 entries Search: linear_reg

MODEL TYPE	ENGINE	PARSNIP	ORIGINAL
All	All	All	All
linear_reg	glmnet	penalty	lambda
linear_reg	glmnet	mixture	alpha
linear_reg	spark	penalty	reg_param
linear_reg	spark	mixture	elastic_net_param
linear_reg	keras	penalty	penalty

ความหมายของการกำหนดอาร์กิวเมนท์แต่ละค่าสามารถศึกษาได้จากคู่มือของฟังก์ชัน `linear_reg()` ซึ่งสามารถกด hyper-link จากคู่มือข้างต้นเข้าไปศึกษาได้เลย (คู่มือ `linear_reg()`)

ตัวอย่างต่อไปนี้แสดงการกำหนดโมเดลด้วย parsnip

```
lm_model <- linear_reg() %>%        # model type
              set_engine("lm") %>%  # model engine
              set_mode("regression") # model mode
```

2. การประมวลผล

เมื่อกำหนดโมเดลแล้วขั้นตอนถัดไปคือการนำ model specification ดังกล่าวไปดำเนินการประมวลผล โดยส่งผ่าน model specification ดังกล่าวไปยังฟังก์ชัน `fit()` ซึ่งมีอาร์กิวเมนท์สำคัญ 2 ตัวได้แก่ model formula และ training dataset

การเขียน model formula จะเขียนอยู่ในรูปของ `y ~ x1+x2+x3+...` โดยที่ y คือตัวแปรตาม ส่วน `x1, x2, x3 ,...` คือตัวแปรอิสระภายในชุดข้อมูลฝึกหัด และสัญลักษณ์ ~ หมายความว่า “regress on” ตัวอย่างต่อไปนี้แสดงการส่งผ่าน model specification `lm_model` ในข้างต้นไปประมวลผล

```
fit_lm <- lm_model %>%
            fit(hwy ~ cty,  # model formula
                data = train_str) # training dataset
```

3. การเรียกดูค่าประมาณพารามิเตอร์ (trained model)

จริง ๆ ผู้วิเคราะห์อาจพิมพ์ชื่อของตัวแปรที่เก็บผลลัพธ์จากการประมวลผลในข้อ 2. โดยตรงก็ได้ เช่น

```
fit_lm
```

```
## parsnip model object
##
##
## Call:
## stats::lm(formula = hwy ~ cty, data = data)
##
## Coefficients:
## (Intercept)          cty
##       1.039        1.329
```

อย่างไรก็ตาม tidymodels มีฟังก์ชัน `tidy()` ซึ่งช่วยสร้างตารางสรุปผลลัพธ์จากการประมาณค่าพารามิเตอร์หรือการเรียนรู้ของโมเดลทำนายที่ใช้ให้อยู่ในรูปแบบเดียวกัน ดังนี้

```
tidy(fit_lm)
```

```
## # A tibble: 2 x 5
##   term        estimate std.error statistic  p.value
##   <chr>          <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)     1.04    0.513       2.03 4.42e- 2
## 2 cty             1.33    0.0295     45.1  1.82e-96
```

ผลลัพธ์ในข้างต้นจะเห็นว่าฟังก์ชัน `tidy()` ได้ดึง output ต่าง ๆ ที่เกี่ยวข้องกับค่าพารามิเตอร์ในโมเดลทำนายออกมาแล้วนำเสนอในรูปแบบตาราง ในกรณีของ linear regression ผลลัพธ์ที่นำเสนอประกอบด้วย ค่าประมาณพารามิเตอร์ของจุดตัดแกนและความชัน และค่าสถิติของค่าประมาณทั้งสองได้แก่ ค่าคลาดเคลื่อนมาตรฐาน ค่าสถิติที และ p-value

หมายเหตุ : ตารางสรุปที่ได้จาก `tidy()` ข้างต้นเป็นตารางประเภท tibble ใน R

4. การทำนาย (prediction)

การนำโมเดลที่ผ่านการ train เรียบร้อยแล้วไปใช้ทำนายสามารถทำได้โดยส่งผ่านโมเดลดังกล่าวซึ่งในที่นี้คือ `fit_lm` ไปยังฟังก์ชัน `predict()` ที่มีอาร์กิวเมนท์สำคัญคือ `new_data` ที่ใช้สำหรับระบุชุดข้อมูลทดสอบ (test data) หรือชุดข้อมูลใหม่ที่มีหน่วยข้อมูลที่ต้องการให้โมเดลทำนาย ตัวอย่างต่อไปนี้แสดงการทำนายค่าสังเกตใน `test_str` ด้วยโมเดล `fit_lm`

```
hwy_pred <- fit_lm %>%
              predict(new_data = test_str)
hwy_pred
```

```
## # A tibble: 62 x 1
##    .pred
##    <dbl>
##  1  21.0
##  2  18.3
##  3  17.0
##  4  22.3
##  5  21.0
##  6  15.7
##  7  30.3
##  8  25.0
##  9  23.6
## 10  25.0
## # ... with 52 more rows
```

ผลลัพธ์ที่ได้จากการทำนายจะเป็นตารางแบบ tibble ที่แต่ละ row คือค่าทำนายของหน่วยข้อมูลใน row เดียวกันกับใน `test_str`

เอกสารอ่านเพิ่มเติม

- https://rawgit.com/pbiecek/DALEX_docs/master/vignettes/DALEX_and_keras.html