บทที่ 7 Regression Models

หัวข้อหลัก

- การเตรียมข้อมูลเพื่อการสร้างโมเดล
- การสร้างโมเดล Linear Regression และ Logistic Regression โดยใช้ไลบารี่ scikit-learn
- ศึกษาวิธีการสกัดฟีเจอร์ที่สำคัญจากโมเดลที่ปรับแต่งแล้ว
- ศึกษาวิธีการประเมินประสิทธิภาพของโมเดล
- การปรับแต่งค่าพารามิเตอร์ด้วยวิธีการค้นหาแบบกริด (grid search)

7.1 Linear Regression

ในการทำนายค่าด้วยวิธี Regression ค่าของตัวแปรตาม (dependent variable) หนึ่งตัวจะถูกทำนายจากค่าของตัวแปรต้น (independent variables) หลายๆ ตัว เราสามารถใช้ Regression สำหรับการทำนายค่า ดังเช่น

- การทำนายเปอร์เซ็นต์ที่ทีมฟุตบอลลิเวอร์พูลจะชนะ เมื่อกำหนดสถิติการเล่นของทีมลิเวอร์พูลและทีมคู่แข่ง
- การทำนายอัตราเสี่ยงของการเกิดหัวใจวาย เมื่อทราบประวัติสุขภาพของบุคคลในครอบครัวและค่าที่วัดได้ทางกายภาพและสรีรวิทยาต่างๆ
- โอกาสที่หิมะจะตก เมื่อกำหนดผลการวัดค่าทางภูมิอากาศต่างๆ

Regression เป็นที่นิยมใช้เนื่องมาจากความง่าย ความโปร่งใส (transparency) ความสามารถในการตีความผลลัพธ์ที่ได้จากโมเดล (interpretability) และความสามารถในการนำไปใช้กับค่าอินพุทที่ไม่ได้อยู่ในชุดข้อมูลฝึกฝน (extrapolation or generalization) ผลลัพธ์ของ linear regression คือเส้นตรงที่ลากผ่านชุดข้อมูลที่ทำให้ค่าของผลต่างระหว่างค่าของจุดข้อมูล (observations) กับค่าของฟังก์ชันเส้นตรง (predicted values) มีค่าน้อยที่สุด

สมมติฐานของ linear regression คือ ความสัมพันธ์ระหว่างฟีเจอร์และตัวแปรตาม สอดคล้องกับสมการเส้นตรง ซึ่งนิยามโดยค่าความชันและจุดตัด (slope and intercept) ในรูปแบบ $y=\alpha+\beta x$ เมื่อ α คือค่าจุดตัด (ค่า *y* เมื่อ *x=0*) และ β คือค่าความชัน ส่วน *x* คือตัวแปรต้น (independent variables)

การสร้าง Simple Linear Regression Model ด้วย Scikit-Learn

1. ดาวน์โหลดข้อมูลอากาศในเมืองๆ หนึ่งของประเทศฮังการี ระหว่าง April 1, 2006 ถึง September 9, 2016

```
In [4]: !wget https://raw.githubusercontent.com/TrainingByPackt/Data-Science-with-Python/master/Chapter03/weather.csv
        --2019-09-20 11:21:23--  https://raw.githubusercontent.com/TrainingByPackt/Data-Science-with-Python/master/Chapter03/w
        eather.csv
        Resolving raw.githubusercontent.com... 151.101.8.133
        Connecting to raw.githubusercontent.com|151.101.8.133|:443... connected.
        HTTP request sent, awaiting response... 200 OK
        Length: 495176 (484K) [text/plain]
        Saving to: 'weather.csv.1'

        weather.csv.1       100%[===================>] 483.57K  1.61MB/s    in 0.3s

        2019-09-20 11:21:24 (1.61 MB/s) - 'weather.csv.1' saved [495176/495176]
```

ข้อมูลดังกล่าวประกอบด้วย ผลการสังเกต หรือ จุดข้อมูลจำนวน 10,000 เรคอร์ด แต่ละจุดข้อมูลมีค่าฟีเจอร์ 8 ค่าดังนี้คือ

- Temperature_c: อุณหภูมิ หน่วยเป็นองศาเซลเซียส
- Humidity: สัดส่วนของความชื้น

- Wind_Speed_kmh: ความเร็วลมหน่วยเป็น กม. ต่อ ชม. (kilometers per hour)
- Wind_Bearing_Degrees: ทิศทางลมหน่วยเป็นดีกรี ในทิศทางตามเข็มนาฬิกาจากทิศเหนือ
- Visibility_km: ความสามารถในการมองเห็นหน่วยเป็น กิโลเมตร
- Pressure_millibars: ความดันอากาศวัดในหน่วย มิลลิบาร์
- Rain: มีค่าเป็น 1 หากฝนตก และมีค่าเป็น 0 หากหิมะตก

2. ใช้ไลบารี่ pandas นำเข้าข้อมูลจากไฟล์ weather.csv

```
In [8]: import pandas as pd

        df = pd.read_csv('weather.csv')
```

3. ตรวจสอบข้อมูลโดยใช้ df.info( ) และ df.head( )

```
In [10]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 8 columns):
Temperature_c           10000 non-null float64
Humidity                10000 non-null float64
Wind_Speed_kmh          10000 non-null float64
Wind_Bearing_degrees    10000 non-null int64
Visibility_km           10000 non-null float64
Pressure_millibars      10000 non-null float64
Rain                    10000 non-null int64
Description             10000 non-null object
dtypes: float64(5), int64(2), object(1)
memory usage: 625.1+ KB
```

```
In [11]: df.head(5)
```

Out[11]:

	Temperature_c	Humidity	Wind_Speed_kmh	Wind_Bearing_degrees	Visibility_km	Pressure_millibars	Rain	Description
0	-0.555556	0.92	11.2700	130	8.0500	1021.60	0	Cold
1	21.111111	0.73	20.9300	330	16.1000	1017.00	1	Warm
2	16.600000	0.97	5.9731	193	14.9086	1013.99	1	Normal
3	1.600000	0.82	3.2200	300	16.1000	1031.59	1	Cold
4	2.194444	0.60	10.8836	116	9.9820	1020.88	1	Cold

4. จะเห็นได้ว่าคอลัมน์ Description เป็นตัวแปรแบบ Categorical เราสามารถตรวจสอบจำนวนประเภททั้งหมดในคอลัมน์ Description ของ dataframe นี้ได้ดังนี้

```
In [20]: levels = len(pd.value_counts(df['Description']))

         print('There are {} levels in the Description column'.format(levels))
```

```
There are 3 levels in the Description column
```

5. แปลง Categorical value attribute ไปเป็น indicator variables

```
In [23]: df_dummies = pd.get_dummies(df, drop_first=True)

         df_dummies.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 9 columns):
Temperature_c           10000 non-null float64
Humidity                10000 non-null float64
Wind_Speed_kmh          10000 non-null float64
Wind_Bearing_degrees    10000 non-null int64
Visibility_km           10000 non-null float64
Pressure_millibars      10000 non-null float64
Rain                    10000 non-null int64
Description_Normal      10000 non-null uint8
Description_Warm        10000 non-null uint8
dtypes: float64(5), int64(2), uint8(2)
memory usage: 566.5 KB
```

6. สลับตำแหน่งของข้อมูลเพื่อป้องกันผลกระทบจากลำดับข้อมูล โดยใช้คำสั่ง shuffle

```
In [25]: from sklearn.utils import shuffle

         df_shuffled = shuffle(df_dummies, random_state=42)
```

7. แบ่งข้อมูลออกเป็นฟีเจอร์ X และตัวแปรตาม y

```
In [27]: DV = 'Temperature_c'

         X = df_shuffled.drop(DV, axis=1)
         y = df_shuffled[DV]
```

8. แบ่ง X และ y ออกเป็น ชุดข้อมูลฝึกฝน และชุดข้อมูลทดสอบ

```
In [28]: from sklearn.model_selection import train_test_split

         X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                test_size=0.33, random_state=42)
```

9. ฟิตโมเดล Simple Linear Regression กับชุดข้อมูลฝึกฝน: โมเดล Simple Linear Regression เป็นโมเดลที่สร้างความสัมพันธ์ระหว่าง ฟีเจอร์หนึ่งตัว กับ ค่าผลลัพธ์แบบต่อเนื่อง (continuous outcome variable) โดยใช้สมการ $y=\alpha+\beta x$ ในตัวอย่างนี้เราจะเลือกฟีเจอร์ Humidity เป็นตัวแปรต้น

```
In [32]: from sklearn.linear_model import LinearRegression

         model = LinearRegression()
```

```
In [33]: model.fit(X_train[['Humidity']], y_train)

Out[33]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
                  normalize=False)
```

10. ดึงค่าพารามิเตอร์ของโมเดลที่เทรนเรียบร้อยแล้ว

```
In [34]: intercept = model.intercept_

         coefficient = model.coef_
```

```
In [37]: print('Temperature = {0:0.2f} + ({1:0.2f}) x Humidity'.format(
                        intercept, coefficient[0]))

         Temperature = 34.50 + (-30.69) x Humidity
```

11. ทำนายและประเมินประสิทธิภาพของโมเดล โดยใช้วิธีการวัดประสิทธิภาพดังนี้คือ

- **Mean Absolute Error (MAE):** คือ ค่าเฉลี่ยของความต่างสัมบูรณ์ระหว่างค่าที่ทำนายได้กับค่าจริง
- **Mean Squared Error (MSE):** คือ ค่าเฉลี่ยของกำลังสองของความแตกต่างระหว่างค่าที่ทำนายได้กับค่าจริง
- **Root Mean Squred Error (RMSE):** คือ รากที่สองของ MSE
- **R-Squared:** เป็นค่าที่บอกถึงสัดส่วนของความแปรปรวนของตัวแปรตามซึ่งสามารถอธิบายได้โดยโมเดลนี้

```
In [38]: predictions = model.predict(X_test[['Humidity']])

         from sklearn import metrics
         import numpy as np

         metrics_df = pd.DataFrame(
                    {'Metric':
                     ['MAE', 'MSE', 'RMSE', 'R-Squared'],
                     'Value':
                     [metrics.mean_absolute_error(y_test, predictions),
                      metrics.mean_squared_error(y_test, predictions),
                      np.sqrt(metrics.mean_squared_error(y_test, predictions)),
                      metrics.explained_variance_score(y_test, predictions)]
                    }).round(3)

         print(metrics_df)
```

```
      Metric   Value
0        MAE   6.052
1        MSE  56.187
2       RMSE   7.496
3  R-Squared   0.389
```

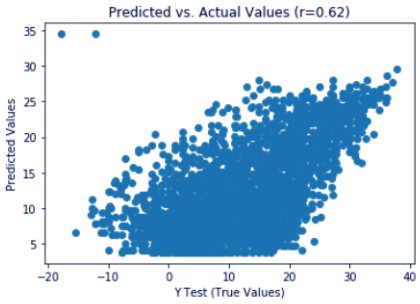
จากตัวอย่างข้างต้นจะพบว่า ค่าความชื้น หรือ Humidity สามารถอธิบายค่าอุณหภูมิหรือตัวแปรตามได้เพียง 38.9% ของค่าความแปรปรวนของอุณหภูมิ และความผิดพลาดของโมเดลจะมีค่าอยู่ภายในช่วง ± 6.052± 6.052

12. ทำความเข้าใจประสิทธิภาพของโมเดลโดยใช้ Data Visualization เราจะเริ่มจากการดูความสัมพันธ์ระหว่างคำทำนายที่ได้จากโมเดลและค่าเอาท์พุทจริง โดยการใช้ scatterplot

In [41]:
```
import matplotlib.pyplot as plt
%matplotlib inline

from scipy.stats import pearsonr

plt.scatter(y_test, predictions)
plt.xlabel('Y Test (True Values)')
plt.ylabel('Predicted Values')
plt.title('Predicted vs. Actual Values (r={0:0.2f})'.format(
                    pearsonr(y_test, predictions)[0], 2))
plt.show()
```



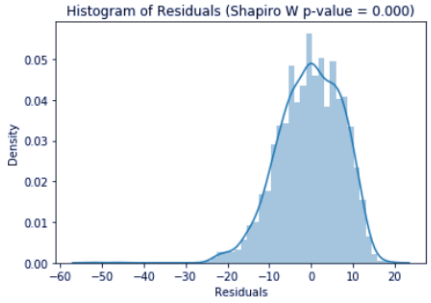
จากกราฟข้างต้นจะเห็นได้ว่า ค่า y_test และ predicted_value แปรผันตามกัน โดยมีค่า Pearson r value เท่ากับ 0.62 ซึ่งหมายความว่า y_test และ predicted_value มีความสัมพันธ์กันเชิงบวกแบบเชิงเส้นในระดับกลาง (moderate, positive linear correlation)

เนื่องจากโมเดลที่ฟิตกับชุดข้อมูลได้ดี จะมีการกระจายของค่าความแตกต่าง (residuals) เป็นแบบปกติ ลำดับถัดไปเราจะทำการพล็อตการกระจายของค่า Residuals เพื่อดูรูปแบบการกระจายว่าเป็นแบบปกติหรือไม่

In [42]:
```
import seaborn as sns
from scipy.stats import shapiro

sns.distplot((y_test - predictions), bins=50)
plt.xlabel('Residuals')
plt.ylabel('Density')
plt.title('Histogram of Residuals (Shapiro W p-value = {0:0.3f})'.format(
            shapiro(y_test - predictions)[1]))

plt.show()
```



จากภาพฮิสโตแกรม จะเห็นได้ว่า Residuals มีการกระจายเบี่ยงไปทางขวา (negatively skewed) และมีค่า Shapiro W p-value เท่ากับศูนย์ หมายความว่าการกระจายข้อมูลนี้ไม่ใช่แบบปกติ (not normal distribution) ดังนั้นโมเดลนี้จึงยังไม่สามารถฟิตกับชุดข้อมูลได้ดีพอ เราจึงจำเป็นต้องพัฒนาโมเดลให้ดีกว่านี้ได้

7.2 Multiple Linear Regression

โมเดล Multiple Linear Regression เป็นโมเดลที่แสดงความสัมพันธ์ระหว่างฟีเจอร์ตั้งแต่สองตัวขึ้นไป กับ ค่าผลลัพธ์แบบต่อเนื่อง (continuous outcome variable) โดยใช้สมการในรูปแบบ $y=\alpha+\beta_1 x_{i,1}+\beta_2 x_{i,2}+...+\beta_p x_{i,p}$

ตัวอย่าง สคริปต์ภาษาไพธอนสำหรับการสร้าง Multiple Linear Regression Model ด้วยไลบารี่ scikit-learn แสดงดังรูปต่อไปนี้ สำหรับรายละเอียดสามารถศึกษาได้จาก Jupyter Notebook ที่แจกให้ผ่านทาง GitHub ของรายวิชา

```
In [44]: # Construct a model

         from sklearn.linear_model import LinearRegression

         model = LinearRegression()
```

```
In [45]: # Fit the Model

         model.fit(X_train, y_train)
```

```
Out[45]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
                  normalize=False)
```

```
In [46]: # Extract intercept, and coefficients

         intercept = model.intercept_
         coefficients = model.coef_

         print('Temperature = {0:0.2f} + ({1:0.2f} x Humidity) + ({2:0.2f} x Wind Speed) + ({3:0.2f} x Wind Bearing Degrees) + (
                   format(intercept,
                   coefficients[0],
                   coefficients[1],
                   coefficients[2],
                   coefficients[3],
                   coefficients[4],
                   coefficients[5],
                   coefficients[6],
                   coefficients[7]))
```

```
Temperature = 3.54 + (-7.93 x Humidity) + (-0.07 x Wind Speed) + (0.00 x Wind Bearing Degrees) + (0.06 x Visibility) +
(0.00 x Pressure) + (5.61 x Rain) + (8.54 x Normal Weather) + (19.10 x Warm Weather)
```

การประเมินประสิทธิภาพโดยใช้ค่า MAE, MSE, RMSE, และ R-Squared แสดงในรูปถัดไป จะเห็นได้ว่า โมเดลที่ได้มีประสิทธิภาพสูงกว่า Simple Linear Regression Model โดยมีค่า MAE เพียง 2.861 และค่า R-Squared เท่ากับ 0.866

```
In [48]: # predict and evaluate
         predictions = model.predict(X_test)

         from sklearn import metrics
         import numpy as np

         metrics_df = pd.DataFrame(
                     {'Metric':
                      ['MAE', 'MSE', 'RMSE', 'R-Squared'],
                      'Value':
                      [metrics.mean_absolute_error(y_test, predictions),
                       metrics.mean_squared_error(y_test, predictions),
                       np.sqrt(metrics.mean_squared_error(y_test, predictions)),
                       metrics.explained_variance_score(y_test, predictions)]
                     }).round(3)

         print(metrics_df)
```

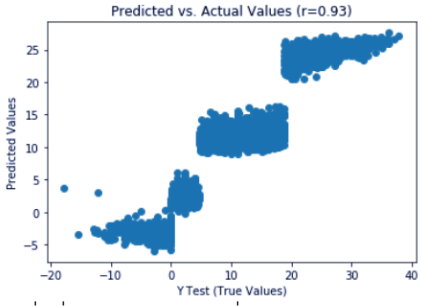
```
      Metric   Value
0        MAE   2.861
1        MSE  12.317
2       RMSE   3.510
3  R-Squared   0.866
```

จากนั้นทำการประเมินคุณภาพของโมเดลโดยใช้ Data Visualization เริ่มจากการพล็อตค่าเอาท์พุทจริงกับคำทำนายของโมเดลบน scatter plot

In [23]:
```
import matplotlib.pyplot as plt
%matplotlib inline

from scipy.stats import pearsonr

plt.scatter(y_test, predictions)
plt.xlabel('Y Test (True Values)')
plt.ylabel('Predicted Values')
plt.title('Predicted vs. Actual Values (r={0:0.2f})'.format(
                    pearsonr(y_test, predictions)[0], 2))
plt.show()
```



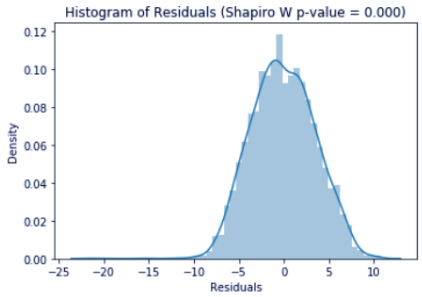
จะเห็นได้ว่า ค่าที่โมเดลทำนายกับค่าจริง มีความสัมพันธ์เชิงเส้นแบบบวกที่เข้มแข็งกว่าในกรณีของ simple linear regression model โดยพบว่าค่า pearson r มีค่าเท่ากับ 0.93

ต่อไปทำการตรวจสอบการกระจายตัวของค่า Residuals ซึ่งพบว่า มีการกระจายตัวแบบไม่ปกติ แต่มีความเบี่ยงเบน (skewness) ที่น้อยกว่าค่า Residuals ในกรณีของ Simple Linear Regression Model

In [27]:
```
import seaborn as sns
from scipy.stats import shapiro

sns.distplot((y_test - predictions), bins=50)
plt.xlabel('Residuals')
plt.ylabel('Density')
plt.title('Histogram of Residuals (Shapiro W p-value = {0:0.3f})'.format(
           shapiro(y_test - predictions)[1]))

plt.show()
```



7.3 Logistic Regression

โมเดลวิเคราะห์การถดถอยแบบโลจิสติก ใช้ค่าของตัวแปรต้นแบบต่อเนื่อง (continuous) และแบบจัดประเภท (categorical) เพื่อทำนายค่าผลลัพธ์ที่มีชนิดเป็นแบบจัดประเภท (categorical outcome) ในกรณีที่ค่าผลลัพธ์มีค่าที่เป็นไปได้สองค่า เราจะเรียกว่า binary logistic regression ส่วนกรณีที่ค่าผลลัพธ์ที่เราต้องการทำนายมีค่าที่เป็นไปได้มากกว่าสองค่า เราจะเรียกว่า multinomial logistic regression

หลักการของการวิเคราะห์การถดถอยแบบโลจิสติก คือการแปลงค่าผลลัพธ์ของการวิเคราะห์การถดถอยแบบเชิงเส้น (linear regression) ให้อยู่ในช่วง (0,1] โดยใช้การแปลงแบบลอการิทึม (logarithmic transformation) สมการของโมเดล logistic regression สำหรับกรณีที่มีตัวแปรต้นเพียงหนึ่งตัว สามารถเขียนได้ดังนี้

$$P(Y) = \frac{1}{1 + e^{-(\alpha+\beta x)}}$$

เมื่อ P(Y) คือค่าความน่าจะเป็นของการเกิดขึ้นของผลลัพธ์ Y, *e* คือ ฐานของลอการิทึมธรรมชาติ, α คือค่าจุดตัด (ค่า *y* เมื่อ *x=0*) และ β คือค่าความชัน ส่วน *x* คือตัวแปรต้น (independent variables) จะเห็นได้ว่าค่าเลขยกกำลัง $\alpha+\beta x$ ก็คือ ค่าผลลัพธ์ของการวิเคราะห์การถดถอยแบบเชิงเส้นนั่นเอง

ในกรณีที่ตัวแปรต้นของการทำนายมีหลายตัวแปร สมการของโมเดลวิเคราะห์การถดถอยแบบโลจิสติก จะอยู่ในรูปดังนี้

$$P(Y) = \frac{1}{1 + e^{-(\alpha+\beta_1 x_{i1} + \beta_2 x_{i2} +\ldots+ \beta_{p-1} x_{ip-1})}}$$

จะเห็นได้ว่า ในการวิเคราะห์การถดถอยแบบเชิงเส้น (linear regression) นั้น เราตั้งสมมติฐานว่ามีความสัมพันธ์ระหว่างตัวแปรต้นและตัวแปรตามจะเป็นแบบเชิงเส้น แต่ในการวิเคราะห์การถดถอยแบบโลจิสติกนั้น สมมติฐานของโมเดลก็คือความสัมพันธ์เชิงเส้นระหว่างค่าของตัวแปรต้นกับค่า natural logarithm ของ *p/(1-p)* เมื่อ *p* คือค่าความน่าจะเป็นของการเกิดขึ้นของผลลัพธ์ (ค่า *P(Y)*)

ต่อไปจะแสดงตัวอย่างการใช้โมเดลวิเคราะห์การถดถอยแบบโลจิสติก เพื่อทำนายความน่าจะเป็นของการเกิดฝนตก โดยใช้ข้อมูลสภาพภูมิอากาศของเมืองหนึ่งในประเทศฮังการีระหว่าง April 1, 2006 ถึง September 9, 2016 ที่ถูกเก็บไว้ในไฟล์ weather.csv จากการวิเคราะห์ในหัวข้อ 7.1 และ 7.2

สคริปต์สำหรับสร้างโมเดล logistic regression ด้วยไลบารี่ scikit-learn

In [ ]:
```
import pandas as pd
from sklearn.utils import shuffle
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression


df = pd.read_csv('weather.csv')
df_dummies = pd.get_dummies(df, drop_first=True)
df_shuffled = shuffle(df_dummies, random_state=42)

# split data into features X and outcome y
DV = 'Rain'
X = df_shuffled.drop(DV, axis=1)
y = df_shuffled[DV]

# split data into training and testing datasets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)

# build a logistic regression model
model = LogisticRegression()
model.fit(X_train, y_train)
```

ค่าพารามิเตอร์ที่เทรนได้มีดังนี้คือ

ค่าพารามิเตอร์ β คือ

In [17]:
```
intercept = model.intercept_

print(intercept[0])
```

```
-0.1775235755667482
```

ค่าพารามิเตอร์ $\alpha_1 \ldots \alpha_8$ คือ

In [45]:
```
coef_df = pd.DataFrame(
    {'Feature': list(X_train.columns), 'Coefficient' : coef_list})

print(coef_df)
```

```
                Feature  Coefficient
0         Temperature_c     5.691326
1              Humidity    -0.165325
2        Wind_Speed_kmh    -0.067057
3  Wind_Bearing_degrees    -0.002367
4         Visibility_km     0.055192
5    Pressure_millibars     0.000845
6    Description_Normal     0.029056
7      Description_Warm     0.001911
```

ประเมินประสิทธิภาพของโมเดล

In [14]:
```
predicted_prob = model.predict_proba(X_test)[:, 1]
predicted_class = model.predict(X_test)


from sklearn.metrics import confusion_matrix
import numpy as np
cm = pd.DataFrame(confusion_matrix(y_test, predicted_class))
cm['Total'] = np.sum(cm, axis=1)
cm = cm.append(np.sum(cm, axis=0), ignore_index=True)
cm = cm.set_index([['Actual No', 'Actual Yes', 'Total']])
print(cm)

print()

from sklearn.metrics import classification_report
print(classification_report(y_test, predicted_class))
```

```
              0     1  Total
Actual No   377     6    383
Actual Yes   10  2907   2917
Total       387  2913   3300

              precision    recall  f1-score   support

           0       0.97      0.98      0.98       383
           1       1.00      1.00      1.00      2917

   micro avg       1.00      1.00      1.00      3300
   macro avg       0.99      0.99      0.99      3300
weighted avg       1.00      1.00      1.00      3300
```

7.4 การปรับแต่งค่าไฮเปอร์พารามิเตอร์ของโมเดลด้วยการค้นหาแบบกริด

จากหัวข้อที่ 7.3 จะเห็นได้ว่าโมเดลของเราสามารถทำนายการเกิดฝนตกได้แม่นยำมาก แต่ยังมีความผิดพลาดอยู่ในกรณีที่ฝนไม่ตก (f1-score = 0.98) ต่อไปเราจะทำการเพิ่มประสิทธิภาพของโมเดลด้วยวิธีการ Grid Search ซึ่งเป็นการค้นหาค่าไฮเปอร์พารามิเตอร์ (hyperparameters) ที่เหมาะสมจากค่าที่เป็นไปได้ โดยค่าไฮเปอร์พารามิเตอร์ของ Logistic Regression ที่เราต้องการปรับแต่งมีดังนี้คือ

- **penalty** : ใช้ระบุค่า norm ที่ใช้สำหรับ penaalization เช่น 'l1' และ 'l2'
- **C** : ค่าอินเวอร์สของสเกลการทำ regularization หากค่า C มีค่าน้อยจะทำให้ regularization term มีค่ามาก
- **solver** : อัลกอริทึมสำหรับ optimization เช่น liblinear, saga, newton-cg

การปรับแต่งโมเดล logistic regression ด้วยวิธีการค้นหาแบบกริดโดยใช้ไลบารี่ scikit-learn แสดงดังซอร์สโค้ดตัวอย่างต่อไปนี้

1. สร้าง grid search modle เพื่อค้นหาค่าไฮเปอร์พารามิเตอร์ที่ทำให้โมเดลมีค่า f1-score สูงที่สุด

```
In [15]: import numpy as np

         grid = {'penalty': ['l1', 'l2'],
                 'C': np.linspace(1, 10, 10),
                 'solver' : ['liblinear']}

         from sklearn.model_selection import GridSearchCV
         from sklearn.linear_model import LogisticRegression

         model = GridSearchCV(LogisticRegression(
                     solver='liblinear'), grid, scoring='f1', cv=5)
```

2. ฟิตโมเดลกับชุดข้อมูลฝึกฝน

```
In [16]: model.fit(X_train, y_train)

Out[16]: GridSearchCV(cv=5, error_score='raise-deprecating',
                estimator=LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, max_iter=100, multi_class='warn',
                   n_jobs=None, penalty='l2', random_state=None, solver='liblinear',
                   tol=0.0001, verbose=0, warm_start=False),
                fit_params=None, iid='warn', n_jobs=None,
                param_grid={'penalty': ['l1', 'l2'], 'C': array([ 1.,  2.,  3.,  4.,  5.,  6.,  7.,  8.,  9., 10.]), 'solver':
             ['liblinear']},
                pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
                scoring='f1', verbose=0)
```

3. ค่าไฮเปอร์พารามิเตอร์ที่ทำให้โมเดลมีค่า f1-score สูงที่สุดคือ

```
In [18]: best_parameters = model.best_params_

         print(best_parameters)

         {'C': 9.0, 'penalty': 'l1', 'solver': 'liblinear'}
```

4. ประเมินประสิทธิภาพของโมเดลที่ได้จากการปรับแต่งค่าไฮเปอร์พารามิเตอร์ด้วยวิธีการ Grid Search

In [19]:
```
predicted_prob = model.predict_proba(X_test)[:, 1]
predicted_class = model.predict(X_test)


from sklearn.metrics import confusion_matrix
import numpy as np
cm = pd.DataFrame(confusion_matrix(y_test, predicted_class))
cm['Total'] = np.sum(cm, axis=1)
cm = cm.append(np.sum(cm, axis=0), ignore_index=True)
cm = cm.set_index([['Actual No', 'Actual Yes', 'Total']])
print(cm)

print()

from sklearn.metrics import classification_report
print(classification_report(y_test, predicted_class))
```

```
              0     1  Total
Actual No   379     4    383
Actual Yes    5  2912   2917
Total       384  2916   3300

              precision    recall  f1-score   support

           0       0.99      0.99      0.99       383
           1       1.00      1.00      1.00      2917

   micro avg       1.00      1.00      1.00      3300
   macro avg       0.99      0.99      0.99      3300
weighted avg       1.00      1.00      1.00      3300
```

จะเห็นได้ว่าโมเดลที่ได้จากการทำ Grid Search กับค่าที่เป็นไปได้ของไฮเปอร์พารามิเตอร์ penalty, C, และ solver มีค่า f1-score ดีกว่าโมเดลก่อนหน้าซึ่งไม่มีการปรับแต่งค่าไฮเปอร์พารามิเตอร์ สำหรับผู้ที่สนใจศึกษารายละเอียดของการปรับแต่งค่าไฮเปอร์พารามิเตอร์ (ไฮเปอร์พารามิเตอร์ คือ คุณลักษณะของโมเดลที่ไม่เกี่ยวข้องกับการทำงานภายในของโมเดล และไม่สามารถประมาณค่าจากชุดข้อมูลได้) สามารถศึกษาเพิ่มเติมได้จากคู่มือของไลบารี่ scikit-learn หรือจาก [3]

แบบฝึกหัด

1. จงสร้างโมเดล linear regression สำหรับทำนายค่า miles-per-gallon ของรถยนต์ โดยใช้ชุดข้อมูล Auto MPG dataset จาก UCI Machine Learning Repository: https://archive.ics.uci.edu/ml/datasets/auto+mpg
2. จงสร้างโมเดล logistic regression สำหรับทำนายค่า miles-per-gallon ของรถยนต์ โดยใช้ชุดข้อมูล Auto MPG dataset จาก UCI Machine Learning Repository: https://archive.ics.uci.edu/ml/datasets/auto+mpg
3. จงอธิบายเปรียบเทียบคุณลักษณะของ linear regression กับ logistic regression
4. จงศึกษาค้นคว้าข้อมูลเกี่ยวกับการปรับแต่งค่าไฮเปอร์พารามิเตอร์ของโมเดลด้วยวิธีการค้นหาแบบกริด (grid search hyperparameter tuning)

เอกสารอ้างอิง

[1] Joel Grus. Data Science from Scratch (2ed), O'Reilly Media, Inc., 2019.

[2] Aaron England; Mohamed Noordeen Alaudeen; Rohan Chopra. Data Science with Python, Packt Publishing, 2019

[3] https://towardsdatascience.com/grid-search-for-model-tuning-3319b259367e