IT525 (2/2017): Homework-01 (50 คะแนน)

Assignment Date: Feb 19, 2018/Submission Date: Mar 4, 2018 before 23:55pm

1 การบ้าน

วัตถุประสงค์: เพื่อให้นิสิตได้ศึกษาและฝึกปฏิบัติงานเกี่ยวกับกระบวนการในวิทยาการข้อมูล

งาน: ให้นิสิตทำการวิเคราะห์หาสมการถดถอยเชิงเส้นชนิดหลายตัวแปร (multiple linear regression) โดยใช้ชุดข้อมูล `hw01_burger_king_nutrition_fact_sheet.csv` (`https://www.bk.com/pdfs/nutrition.pdf`) เพื่อแสดงความสัมพันธ์ระหว่างปริมาณแคลอรี่ (Calories) และสารอาหารต่างๆ โดยใช้แนวทางที่ให้ดังต่อไปนี้

- เขียนโปรแกรมภาษาไพทอน ใน Jupyter Notebook ตามลำดับข้อและคำสั่งที่ให้มาด้านล่าง พร้อมทั้งแสดงผลลัพธ์
- ส่วนที่เป็นช่องสีแดงคือ สิ่งที่นิสิตต้องคิดเอง และส่วนที่เป็น source code สีแดงเป็นสิ่งที่คิดไว้ให้แล้ว ซึ่งนิสิตสามารถนำไปพิมพ์ใน Jupyter Notebook ได้เลย
- ส่งไฟล์ .ipynb บน ATutor โดยใช้ชื่อไฟล์ `it525-hw01-your_student_number.ipynb`

2 Data Wrangling and Pre-processing

1. ทำการอิมพอร์ตเพ็กเก็จและกำหนดค่าต่างๆที่จำเป็น

2. เปลี่ยนไดเร็กทอรี่ไปที่ไดเร็กทอรี่ที่มีไฟล์ชุดข้อมูลอยู่ และทำการอ่านไฟล์ชุดข้อมูลไปเก็บไว้ในตัวแปรชื่อ `df`

3. แสดงจำนวนแถวและจำนวนคอลัมน์ของชุดข้อมูล

4. แสดงรายชื่อคอลัมน์ที่มีอยู่ในชุดข้อมูล

5. ทำการลบช่องว่างที่อยู่หน้าชื่อคอลัมน์ออก เพื่อว่าเวลาอ้างถึงชื่อคอลัมน์จะได้ใช้ชื่อตรงๆได้เลยไม่ต้องพิมพ์ช่องว่าง

```
>>>df.columns = map(lambda x: x.strip(), df.columns.values)
df.columns
Index(['Nutrition facts', 'serving size (g)', 'Calories', 'Calories from fat',
       'Total fat (g)', 'Saturated Fat (g)', 'Trans Fat (g)', 'Chol (mg)',
       'Sodium (mg)', 'Total Carb (g)', 'Dietary Fiber (g)', 'Total Sugar (g)',
       'Protein (g)'],
      dtype='object')
```

6. นับจำนวนอาหารในชนิด `burger`, `sandwich`, `salad`, และ `beverage` พร้อมทั้งแสดงภาพโดยใช้ `bar graph` และ `pie chart`

   - นับจำนวนอาหารที่มีอยู่ในแต่ละชนิดที่เป็น `burger`, `sandwich`, `salad`, และ `beverage`

     ```
     >>>mydict = dict()
     >>>mydict['burger'] = len(df[df['Nutrition facts'].str.contains('burger|Burger')])
     >>>mydict['sandwich'] = len(df[df['Nutrition facts'].str
             .contains('Sandwich|Sandwiches')])
     >>>mydict['salad'] = len(df[df['Nutrition facts'].str.contains('Salad|salad')])
     >>>mydict['beverage'] = len(df[df['Nutrition facts'].str.contains('Milk|Juice|Shake|
             Smoothie|Tea|Lemonade|Coke|Coca Cola|Orange|Punch|Sprite|Beer|Pepper')])
     ```

   - สร้าง `bar graph` และ `pie chart` เพื่อแสดงสัดส่วนจำนวนอาหารที่มีอยู่ในแต่ละชนิด โดยให้กราฟทั้งสองอยู่ใน `figure` เดียวกัน แต่คนละ `subplots`

     ```

     ```

7. ทำการแทนค่าข้อมูลใน cell ที่มีแต่ช่องว่าง (whitespace) อย่างเดียวด้วย `NaN` —เนื่องจากข้อมูลในชุดข้อมูลอาจมีข้อมูลที่เป็นช่องว่าง (whitespace) อย่างเดียว อยู่ โดยอาจจะเป็นความตั้งใจหรือความผิดพลาดของการป้อนข้อมูลหรือสร้างชุดข้อมูลขึ้นมา ซึ่งทำให้โปรแกรมคิดว่ามันเป็นข้อมูลชนิดหนึ่ง ไม่ใช่ข้อมูลที่ขาดหาย (missing value)

   - แทนค่าข้อมูลที่เป็นช่องว่างอย่างเดียวด้วย `NaN` เช่น ในไฟล์ csv แทนที่ `,' ',` ด้วย `,NaN,`

     ```
     >>>df.replace(r'^\s*$', np.nan, regex=True, inplace=True)
     ```

   - แสดงจำนวนแถวของข้อมูลที่มีค่า `NaN` อยู่อีกครั้ง

     ```
     >>>len(df[df.isnull().any(axis=1)])
     94
     ```

8. ลบแถวข้อมูลที่มีค่า `NaN` อยู่ ไม่ว่าจะมีแค่คอลัมน์เดียวหรือทุกคอลัมน์ในแถวข้อมูลนั้นๆ

   - แสดงลิสต์ของ row indices ที่มีค่า `NaN`

     ```
     >>>r_index = df[df.isnull().any(axis=1)].index.tolist()
     ```

   - แสดงจำนวนแถวข้อมูลที่มีค่า `NaN` อยู่ ซึ่งควรมี 94 แถว

     ```
     >>>len(r_index)
     94
     ```

   - ทำการลบแถวข้อมูลที่มีค่า `NaN` อยู่

     ```

     ```

   - ตรวจสอบจำนวนแถวของชุดข้อมูลหลังจากลบแถวที่มีค่า `NaN` ออกแล้ว ซึ่งควรเท่ากับ $210 - 94 = 116$

     ```

     ```

9. ตรวจสอบชนิดของข้อมูลในแต่ละคอลัมน์ ใน Pandas DataFrame

   - แสดงชนิดของข้อมูลในแต่ละคอลัมน์

     ```

     ```

     หมายเหตุ: จากผลลัพธ์ ควรจะพบว่าข้อมูลในคอลัมน์ `serving size (g)` และ `Total Carb (g)` เป็นชนิด `object` ซึ่งควรจะเปลี่ยนมันเป็น `float64`

- แปลงข้อมูลในคอลัมน์ `serving size (g)` และ `Total Carb (g)` จากชนิด `object` ไปเป็น `float64`

- ตรวจสอบชนิดของข้อมูลอีกครั้ง หลังจากการแปลงชนิด

```
>>>df.dtypes
Nutrition facts       object
serving size (g)     float64
Calories             float64
Calories from fat    float64
Total fat (g)        float64
Saturated Fat (g)    float64
Trans Fat (g)        float64
Chol (mg)            float64
Sodium (mg)          float64
Total Carb (g)       float64
Dietary Fiber (g)    float64
Total Sugar (g)      float64
Protein (g)          float64
dtype: object
```

10. แสดงค่าสถิติเชิงพรรณาสำหรับข้อมูลใน DataFrame

11. วาด scatter plot แสดงความสัมพันธ์ระหว่างจำนวน serving size (g) และ Calories

12. แปลงค่าปริมาณสารอาหาร ให้อยู่ในอัตราส่วนของ serving size (g) ที่เท่ากัน —เนื่องจากปริมาณสารอาหารในอาหารแต่ละชนิด มาจากปริมาณ serving size (g) ที่ต่างกัน ดังนั้น เราควรจะแปลงค่ามันให้เทียบมาจาก ปริมาณ serving size (g) ที่เท่าๆกัน โดยมีหลักคิดคือ —ถ้า ปริมาณ serving size = s, มีปริมาณสารอาหารในคอลัมน์ B = b ดังนั้น ที่ปริมาณ serving size = max_s จะมีปริมาณสารอาหารในคอลัมน์ $B = b * max\_s/s$

    - หาปริมาณ serving size (g) ที่สูงที่สุดในชุดข้อมูล และเก็บไว้ในตัวแปรชื่อ `max_ss`

    - แปลงค่าปริมาณสารอาหารในคอลัมน์ต่างๆ โดยเทียบกับ serving size (g) ที่สูงที่สุด

```
>>>orig_serving_size = df['serving size (g)'].copy()
>>>df[df.select_dtypes(include=['float64']).columns] =
        df[df.select_dtypes(include=['float64']).columns]*max_ss
>>>df[df.select_dtypes(include=['float64']).columns]
        .div(orig_serving_size.values, axis='rows')
```

13. ตรวจดูว่าข้อมูลมีค่าที่ต่ำหรือสูงผิดปกติ (outliers) หรือไม่

    - วาด boxplot สำหรับข้อมูลทุกคอลัมน์ ยกเว้นคอลัมน์แรกที่ไม่ใช่ข้อมูลตัวเลข โดยให้ boxplots ทุกอันอยู่ใน figure และ plot เดียวกัน

หมายเหตุ: data points ที่อยู่นอก boxplot จะถูกพิจารณาว่าเป็น outliers

- แปลงค่าข้อมูลไปเป็นค่า Z หรือ ค่ามาตรฐาน (standard value) เพื่อดูว่ามีค่าใดบ้างที่ต่ำกว่า -3 หรือสูงกว่า 3 หรือไม่

14. ลบแถวข้อมูลที่มี outliers อยู่

- พิมพ์ลิสต์ของ indice ของแถวของข้อมูลที่มี outliers อยู่ (อ้างอิง: `https://pyformat.info/`)

```
>>>def find_outlier(df, df_zscore):
    for col in df.loc[:, 'serving size (g)':].columns:
        ind = df_zscore.loc[(df_zscore[col]>3) | (df_zscore[col]<-3)]
                .index.tolist()
        print('Column name: {}, Row index: {}'.format(col, ind))

>>>find_outlier(df, df_zscore)
Column name: serving size (g), Row index: []
Column name: Calories, Row index: [66, 124]
Column name: Calories from fat, Row index: [66, 124, 128, 131]
Column name: Total fat (g), Row index: [66, 124, 128, 131]
Column name: Saturated Fat (g), Row index: [107, 122]
Column name: Trans Fat (g), Row index: []
Column name: Chol (mg), Row index: []
Column name: Sodium (mg), Row index: [122]
Column name: Total Carb (g), Row index: [126]
Column name: Dietary Fiber (g), Row index: []
Column name: Total Sugar (g), Row index: [125]
Column name: Protein (g), Row index: []
```

หมายเหตุ: ลองเปรียบเทียบจำนวน outliers ในแต่ละคอลัมน์ กับ boxplot ที่สร้างไว้ก่อนหน้านี้ จะพบว่าจำนวน data points ที่อยู่นอก boxplot จะใกล้เคียงกับจำนวน outliers ที่หาได้ด้วย Z-score

- รวบรวม indice ของแถวข้อมูลที่มี outliers อยู่ และเก็บไว้ในตัวแปรชื่อ `outlier_ind_lst`

```
>>>def collect_outlier_indice(df, df_zscore):
    item_list=[]
    seen = set(item_list)
    for col in df.loc[:, 'serving size (g)':].columns:
        item_list = df_zscore.loc[(df_zscore[col]>3) | (df_zscore[col]<-3)]
                .index.tolist()
        for item in item_list:
            #print(item)
            if item not in seen:
                #print(item)
                seen.add(item)
                #item_list.append(item)
    outlier_indice_lst = list(seen)
    #print(outlier_indice_lst)
    return outlier_indice_lst
    #return seen
```

- เรียกใช้ฟังก์ชัน `collect_outlier_indice()` โดยให้เก็บค่าที่คืนไว้ในตัวแปรชื่อ `outlier_ind_lst`

- ทำการลบแถวข้อมูลที่มี outliers อยู่ออกจากชุดข้อมูล

15. วาด scatter plot เพื่อพิจารณาความสัมพันธ์ระหว่างปริมาณแคลลอรี่ (Calories) และสารอาหารอื่นๆ โดยให้แบ่งเป็น subplots ที่มี 2 แถวและ 5 คอลัมน์

    - Calories และ Calories from fat
    - Calories และ Total fat (g)
    - Calories และ Trans Fats (g)
    - Calories และ Saturated Fat (g)
    - Calories และ Chol (mg)
    - Calories และ Sodium (mg)
    - Calories และ Total Carb (g)
    - Calories และ Dietary Fiber (g)
    - Calories และ Total Sugar (g)
    - Calories และ Protein (g)

16. คำนวณค่าสหสัมพันธ์ (correlation) ระหว่าง Calories และสารอาหารอื่นๆ

    หมายเหตุ: เมื่อพิจารณาจากตารางค่าสหสัมพันธ์ จะพบว่า:

    - Calories from fat, Total fat (g), Saturated Fat (g), Chol (mg), Sodium (mg), Total Carb (g), และ Protein (g) มีค่าสหสัมพันธ์กับ Calories เกินกว่า 0.60
    - Calories from fat และ Total fat (g) มีค่าสหสัมพันธ์กัน 0.99 ซึ่งเสมือนเป็นค่าเดียวกัน
    - ค่าปริมาณไขมัน Total fat (g) สามารถแยกเป็น Trans fat (g) และ Saturated fat (g) ซึ่งเราจะใช้สองค่านี้แทน Total fat (g)
    - Calories from fat และ Total fat (g) มีค่าสหสัมพันธ์กัน 0.99 ซึ่งเสมือนเป็นค่าเดียวกัน ดังนั้นเราจะตัด Calories from fat ออกเช่นกัน

3 Regression Analysis

จากของการทำ data wrangling and pre-processing ในขั้นตอนก่อนหน้านี้ ทำให้เราสามารถสร้างสมการถดถอยชนิดหลายตัวแปรได้ดังนี้

$$Y = b0 + b1 \cdot X1 + b2 \cdot X2 + b3 \cdot X3 + b4 \cdot X4 + b5 \cdot X5 + b6 \cdot X6 + b7 \cdot X7$$

โดยที่

- Y = 'Calories'
- X1 = 'Saturated Fat (g)'
- X2 = 'Trans Fat (g)'
- X3 = 'Chol (mg)'
- X4 = 'Sodium (mg)'
- X5 = 'Total Carb (g)'
- X6 = 'Total Sugar (g)'
- X7 = 'Protein (g)'

1. สร้างสมการถดถอยจากชุดข้อมูล และเก็บไว้ในตัวแปรชื่อ `model1`

2. แสดงค่าสัมประสิทธิ์ของสมการสำหรับ `model1`

3. แสดงข้อมูลทางสถิติของ `model1`

   หมายเหตุ: จากค่าสรุปจะพบว่า ตัวแปรทำนายทุกตัวมีค่า p-value ต่ำกว่า 0.05 และค่า Adjusted R-squared เท่ากับ 0.945 ซึ่งเป็นค่าสูง แต่อย่างไรก็ตาม ค่า p-value ของ intercept เท่ากับ 0.087 ซึ่งสูงกว่า 0.05 ดังนั้นเราจะลองตัดค่าสัมประสิทธิ์ออก โดยบังคับให้สมการถดถอยตัดที่จุดกำเนิด เช่น (0, 0, 0, 0, 0, 0, 0)

4. ทดลองสร้างสมการถดถอย โดยไม่ใช้ค่า intercept —ใช้สมการ $formula = ' y \ X - 1'$ และให้เก็บไว้ในตัวแปรชื่อ `model2`

5. แสดงค่าสัมประสิทธิ์ของสมการสำหรับ `model2`

   หมายเหตุ: ผลลัพธ์จากการตัดค่า intercept ออก พบว่า ค่า p-Value ของตัวแปรทำนายทุกตัวมีค่าต่ำกว่า 0.05 และค่า Adjusted R-squared เพิ่มขึ้นเป็น 0.987 อย่างไรก็ตาม สมการถดถอยที่ไม่มีค่า intercept อยู่จะหมายถึงสมการที่ปริมาณแคลลอรี่ ขึ้นอยู่กับปริมาณสารอาหารในตัวแปรที่กำหนดเท่านั้น แต่ในความเป็นจริงอาจจะขึ้นอยู่กับปริมาณสารอาหารอื่นๆที่ไม่ได้อยู่ในสมการก็ได้ ดังนั้นสมการจึงไม่สะท้อนถึงข้อเท็จจริงเท่าไหร่ —ตรวจดูการประเมินสมรรถนะในเนื้อหาส่วนถัดไป

4 Performance Evaluation

นอกจากค่า p-Value และค่า Adjusted R-squared ที่ใช้สำหรับการบ่งชี้ถึงโมเดลสมการถดถอยที่ดีแล้ว เราสามารถใช้ค่าความคลาดเคลื่อนจากการทำนายในการประเมินสมรรถนะของแบบจำลองเช่นกัน ซึ่งค่าที่รู้จักกันดีคือ Root Mean Squared Error (RMSE)

1. ทำนายค่า Calories จาก `model1` และ `model2`

2. คำนวณค่า root mean squared error ของ `model1` และ `model2`

   หมายเหตุ: จะเห็นว่าเมื่อตัดค่า intercept ออก ถึงแม้ว่าจะทำให้ค่า Adjusted R-squared สูงขึ้น แต่ก็ทำให้ค่า RMSE สูงขึ้นเช่นกัน ในกรณีนี้ สำหรับสมการถดถอยเชิงเส้นชนิดหลายตัวแปร `model1` ดีกว่า `model2` เพราะ `model1` คำนึงถึงค่า intercept และมีค่าความคลาดเคลื่อนจากการทำนาย (RMSE) ต่ำกว่า ถึงแม้ว่าค่า Adjusted R-squared ของมันจะต่ำกว่า `model2`

3. มีแบบจำลองสมการถดถอยที่ดีกว่า `model1` และ `model2` หรือไม่ ในแง่ของการมีค่า RMSE ที่ต่ำกว่าและใช้ intercept ถ้ามี แสดงวิธีการที่ได้มา ผลลัพธ์การคำนวณ (คะแนนพิเศษ 5 คะแนน)