รายงาน

ปัญหา Classification และ Regression

จัดทำโดย

นายศุภณัฐ	บุญสารี	593020466-7
นายพรเทพ	เนตรเดชา	593020930-8

เสนอ

ผศ.ดร.สิรภัทร เชี่ยวชาญวัฒนา

322497 Special Topics in Computer Science (Data Analytics and Mining)

ภาคเรียนที่ 1 ปีการศึกษา 2561

ภาควิชาวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์

มหาวิทยาลัยขอนแก่น

ปัญหา Classification

นำเข้าข้อมูลจาก UCI Machine Learning ที่ชื่อ hepatitis โดยใช้คำสั่ง

```
import pandas as pd
data = pd.read_csv('test.csv')
data
```

วิเคราะห์ความเสียหายของข้อมูล

```
import numpy as np

data = data.replace('?',np.NaN)

print('Number of instances = %d' % (data.shape[0]))
print('Number of attributes = %d' % (data.shape[1]))

print('Number of missing values:')
for col in data.columns:
    print('\t%s: %d' % (col,data[col].isna().sum()))
```

จะได้ผลลัพธ์ดังนี้ :

```
Number of instances = 155
Number of attributes = 20
Number of missing values:
        Class: 0
        AGE: 0
        SEX: 0
        STEROID: 1
        ANTIVIRALS: 0
        FATIGUE: 1
        MALAISE: 1
        ANOREXIA: 1
        LIVER_BIG: 10
        LIVER_FIRM: 11
        SPLEEN_PALPABLE: 5
        SPIDERS: 5
        ASCITES: 5
        VARICES: 5
        BILIRUBIN: 6
        ALK_PHOSPHATE: 29
        SGOT: 4
        ALBUMIN: 16
        PROTIME: 67
        HISTOLOGY: 0
```

จัดการกับข้อมูลโดยการเติมข้อมูลเข้าไปแทนที่ข้อมูลที่เป็น '?' โดยใช้ median เพื่อให้ค่าที่เป็น ? หายไป

```
data2 = data

print('Before replacing missing values:')
print(data2[140:145])
data2 = data2.fillna(data2.median())

print('\nAfter replacing missing values:')
print(data2[140:145])
```

จะได้ผลลัพธ์ดังนี้ :

```
Before replacing missing values:
     Class  AGE  SEX STEROID  ANTIVIRALS FATIGUE MALAISE ANOREXIA LIVER_BIG  \
140      2   36    1       1           2       1       1        1          1
141      1   54    1       1           2       1       1        2        NaN
142      2   51    1       2           2       1       2        2          2
143      1   49    1       1           2       1       1        2          2
144      1   45    1       2           2       1       1        1          2

     LIVER_FIRM SPLEEN_PALPABLE SPIDERS ASCITES VARICES BILIRUBIN  \
140         1              2        1        2       1       1.70
141       NaN              1        2        1       2       3.90
142         1              1        1        2       1       1.00
143         2              1        1        2       2       1.40
144         2              2        1        1       2       1.90

     ALK_PHOSPHATE SGOT ALBUMIN PROTIME  HISTOLOGY
140          295   60     2.7     NaN          2
141          120   28     3.5      43          2
142          NaN   20     3.0      63          2
143           85   70     3.5      35          2
144          NaN  114     2.4     NaN          2

After replacing missing values:
     Class  AGE  SEX STEROID  ANTIVIRALS FATIGUE MALAISE ANOREXIA LIVER_BIG  \
140      2   36    1       1           2       1       1        1          1
141      1   54    1       1           2       1       1        2          2
142      2   51    1       2           2       1       2        2          2
143      1   49    1       1           2       1       1        2          2
144      1   45    1       2           2       1       1        1          2

     LIVER_FIRM SPLEEN_PALPABLE SPIDERS ASCITES VARICES BILIRUBIN  \
140         1              2        1        2       1       1.70
141         2              1        2        1       2       3.90
142         1              1        1        2       1       1.00
143         2              1        1        2       2       1.40
144         2              2        1        1       2       1.90

     ALK_PHOSPHATE SGOT ALBUMIN PROTIME  HISTOLOGY
140          295   60     2.7      61          2
141          120   28     3.5      43          2
142           85   20     3.0      63          2
143           85   70     3.5      35          2
144           85  114     2.4      61          2
```

วิเคราะห์ข้อมูลอีกครั้งเพื่อดูว่าความเสียหายของข้อมูลได้หายไปหรือยัง

```
import numpy as np

data2 = data2.replace('?',np.NaN)

print('Number of instances = %d' % (data2.shape[0]))
print('Number of attributes = %d' % (data2.shape[1]))

print('Number of missing values:')
for col in data2.columns:
    print('\t%s: %d' % (col,data2[col].isna().sum()))
```

จะได้ผลลัพธ์ดังนี้ :

```
Number of instances = 155
Number of attributes = 20
Number of missing values:
        Class: 0
        AGE: 0
        SEX: 0
        STEROID: 0
        ANTIVIRALS: 0
        FATIGUE: 0
        MALAISE: 0
        ANOREXIA: 0
        LIVER_BIG: 0
        LIVER_FIRM: 0
        SPLEEN_PALPABLE: 0
        SPIDERS: 0
        ASCITES: 0
        VARICES: 0
        BILIRUBIN: 0
        ALK_PHOSPHATE: 0
        SGOT: 0
        ALBUMIN: 0
        PROTIME: 0
        HISTOLOGY: 0
```

ค่าเป็น 0 ทั้งหมดถือว่าไม่มี missing value แล้ว

แบ่งข้อมูลเพื่อทำ Training data และ Testing data

```
dups = data2.duplicated()
print('Duplicated rows = %d' % (dups.sum()))

print('Number of rows in original data = %d' % (dups.shape[0]))

data3 = data2.drop_duplicates()

print('Number of rows after discarding duplicated = %d' % (data3.shape[0]))
```

```
Duplicated rows = 0
Number of rows in original data = 155
Number of rows after discarding duplicated = 155
```

```
print('DIE = %d' % (data3['Class'].loc[data3['Class']==1].count()))
print('LIVE = %d' % (data3['Class'].loc[data3['Class']==2].count()))
```

```
DIE = 32
LIVE = 123
```

```
data3_die = data3.loc[data3['Class']==1]
data3_live = data3.loc[data3['Class']==2]
```

```
data3_live_test = data3_live.sample(n=42)
data3_live_test

data4 = data3_die.append(data3_live_test)

data5 = data4.sample(n=74)
data5
```

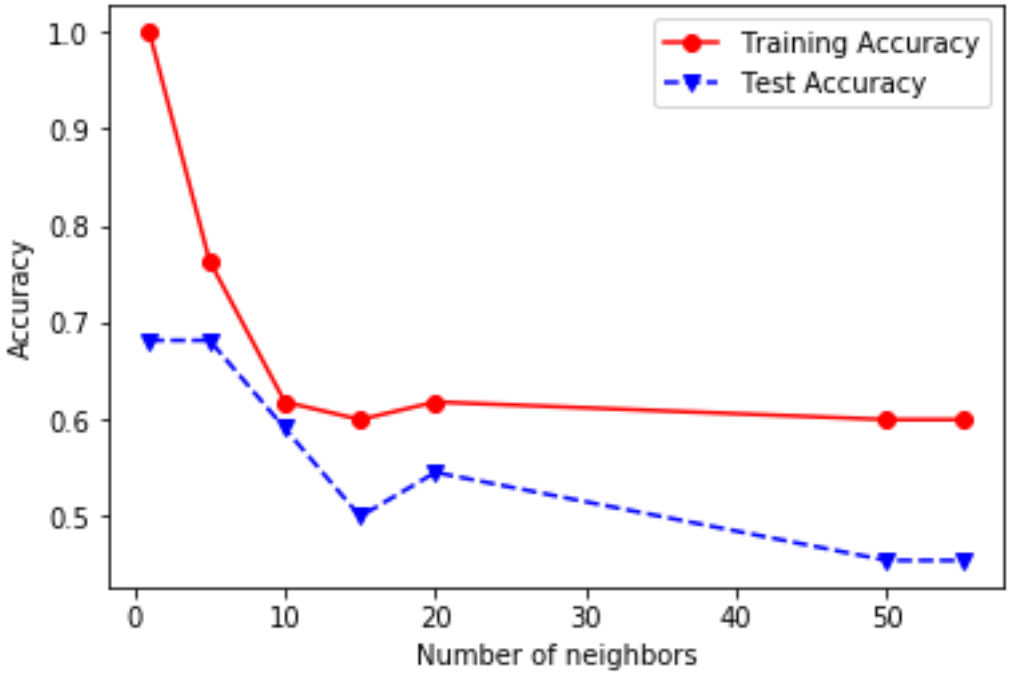
โดยใช้ข้อมูลเพื่อนำไป test = 42 ตัว

จากนั้นทำการ training และ testing data โดยใช้ KNeighborsClassifier และ linear_model

การทำ KNeighborsClassifier

```
from sklearn.neighbors import KNeighborsClassifier
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.metrics import accuracy_score

numNeighbors = [1, 5, 10, 15, 20, 50, 55]
trainAcc = []
testAcc = []

for k in numNeighbors:
    clf = KNeighborsClassifier(n_neighbors=k, metric='minkowski', p=2)
    clf.fit(X_train, Y_train)
    Y_predTrain = clf.predict(X_train)
    Y_predTest = clf.predict(X_test)
    trainAcc.append(accuracy_score(Y_train, Y_predTrain))
    testAcc.append(accuracy_score(Y_test, Y_predTest))

plt.plot(numNeighbors, trainAcc, 'ro-', numNeighbors, testAcc,'bv--')
plt.legend(['Training Accuracy','Test Accuracy'])
plt.xlabel('Number of neighbors')
plt.ylabel('Accuracy')
```

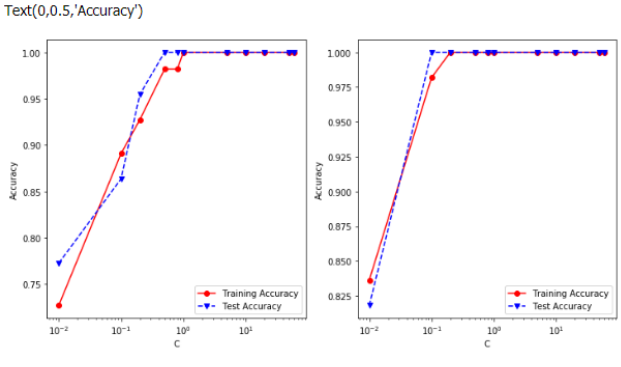
จะได้ผลลัพธ์ดังนี้ :



การทำ linear model

```
from sklearn import linear_model
from sklearn.svm import SVC

C = [0.01, 0.1, 0.2, 0.5, 0.8, 1, 5, 10, 20, 50, 60]
LRtrainAcc = []
LRtestAcc = []
SVMtrainAcc = []
SVMtestAcc = []

for param in C:
    clf = linear_model.LogisticRegression(C=param)
    clf.fit(X_train, Y_train)
    Y_predTrain = clf.predict(X_train)
    Y_predTest = clf.predict(X_test)
    LRtrainAcc.append(accuracy_score(Y_train, Y_predTrain))
    LRtestAcc.append(accuracy_score(Y_test, Y_predTest))

    clf = SVC(C=param,kernel='linear')
    clf.fit(X_train, Y_train)
    Y_predTrain = clf.predict(X_train)
    Y_predTest = clf.predict(X_test)
    SVMtrainAcc.append(accuracy_score(Y_train, Y_predTrain))
    SVMtestAcc.append(accuracy_score(Y_test, Y_predTest))

fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12,6))
ax1.plot(C, LRtrainAcc, 'ro-', C, LRtestAcc,'bv--')
ax1.legend(['Training Accuracy','Test Accuracy'])
ax1.set_xlabel('C')
ax1.set_xscale('log')
ax1.set_ylabel('Accuracy')

ax2.plot(C, SVMtrainAcc, 'ro-', C, SVMtestAcc,'bv--')
ax2.legend(['Training Accuracy','Test Accuracy'])
ax2.set_xlabel('C')
ax2.set_xscale('log')
ax2.set_ylabel('Accuracy')
```

จะได้ผลลัพธ์ดังนี้ :



สรุป

จากภาพข้างต้นจะได้ว่า การ train ข้อมูลโดยการเติมข้อมูลเข้าไป ให้ผลที่มี accuracy ที่สูงที่สุด

```
from sklearn.metrics import accuracy_score

print('Accuracy on test data is %.2f' % (accuracy_score(testY, predY)))
```

```
Accuracy on test data is 1.00
```

ปัญหา regression

จากข้อมูลสภาพอากาศซึ่งทำการตรวจวัดจากสถานี ภาคตะวันออกเฉียงเหนือ จังหวัดขอนแก่น ตั้งแต่ปี 2550 – 2561 โดยนำเข้าข้อมูล 4 ชนิด ดังนี้

1. ก๊าซซัลเฟอร์ไดออกไซด์
2. ก๊าซไนโตรเจนไดออกไซด์
3. ก๊าซคาร์บอนมอนอกไซด์
4. ก๊าซโอโซน

```
import pandas as pd
df = pd.read_csv('air.csv')
df
```

จะได้ผลลัพธ์ดังนี้ :

	ก๊าซซัลฟอร์ไดออกไซด์	ก๊าซไนโตรเจนออกไซด์	ก๊าซคาร์บอนมอนอกไซด์ 8 ชั่วโมง	ก๊าซโอโซน 1 ชั่วโมง
0	3.2	28.3	1.30	13.4
1	5.2	37.2	1.40	22.4
2	3.0	23.0	1.10	25.4
3	4.3	21.5	0.80	21.6
4	3.3	12.6	1.00	4.6
5	2.8	7.7	0.70	11.7
6	2.3	10.3	0.60	12.2
7	1.5	10.3	0.70	9.2
8	1.3	12.7	0.70	10.7
9	2.8	15.9	0.90	16.9
10	3.2	21.8	0.90	18.6
25	3.0	29.0	1.10	18.0
26	2.0	23.0	0.80	24.0
27	3.0	20.0	1.00	24.0
28	2.0	16.0	0.60	23.0
29	1.0	12.0	0.40	19.0
...	...	...	...	...
114	NaN	NaN	NaN	NaN
115	2.0	7.0	0.21	NaN
116	2.0	9.0	0.44	NaN
117	2.0	12.0	0.80	NaN
118	3.0	10.0	0.61	NaN
119	3.0	11.0	0.67	NaN
120	1.0	7.0	NaN	NaN

วิเคราะห์ความสูญหายของข้อมูล

```
import numpy as np

print('Number of instances = %d' % (df.shape[0]))
print('Number of attributes = %d' % (df.shape[1]))

print('\nNumber of missing values:')
for col in df.columns:
    print('\t%s: %d' % (col,df[col].isna().sum()))
```

```
Number of instances = 144
Number of attributes = 4

Number of missing values:
	ก๊าซซัลฟอร์ไดออกไซด์  : 7
	ก๊าซไนโตรเจนออกไซด์: 9
	ก๊าซคาร์บอนมอนอกไซด์ 8 ชั่วโมง: 26
	ก๊าซโอโซน 1 ชั่วโมง: 36
```

จัดการกับข้อมูลโดยการตัดทิ้งออกไป

```
df.dropna(inplace=True)
```

```
df
```

	ก๊าซซัลฟอร์ไดออกไซด์	ก๊าซไนโตรเจนออกไซด์	ก๊าซคาร์บอนมอนอกไซด์ 8 ชั่วโมง	ก๊าซโอโซน 1 ชั่วโมง
0	3.2	28.3	1.30	13.4
1	5.2	37.2	1.40	22.4
2	3.0	23.0	1.10	25.4
3	4.3	21.5	0.80	21.6
4	3.3	12.6	1.00	4.6
5	2.8	7.7	0.70	11.7
6	2.3	10.3	0.60	12.2
7	1.5	10.3	0.70	9.2
8	1.3	12.7	0.70	10.7
9	2.8	15.9	0.90	16.9
10	3.2	21.8	0.90	18.6
11	3.0	21.1	1.00	14.3
12	2.6	24.3	1.40	16.8
13	1.7	23.0	0.90	22.3
14	2.4	24.4	0.80	23.5
15	2.0	17.8	0.60	18.9
16	0.9	12.1	0.50	16.3
17	1.2	11.8	0.40	13.0
18	2.4	12.2	0.40	10.7
19	3.0	11.4	0.40	13.0
20	1.9	15.2	0.70	8.6

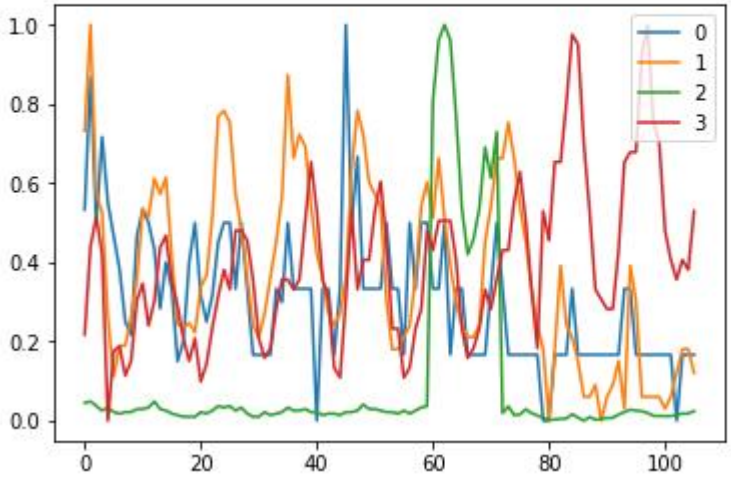
แปลงข้อมูลให้อยู่ในรูปอย่างง่ายเพื่อสามารถนำไปสร้างตัวแบบ ทำการแปลงข้อมูลโดยการใช้ normalize ให้อยู่ในช่วง 0 – 1

```
df_normalized
```

	0	1	2	3
0	0.533333	0.731928	0.044857	0.217822
1	0.866667	1.000000	0.048724	0.440594
2	0.500000	0.572289	0.037123	0.514851
3	0.716667	0.527108	0.025522	0.420792
4	0.550000	0.259036	0.033256	0.000000
5	0.466667	0.111446	0.021655	0.175743
6	0.383333	0.189759	0.017788	0.188119
7	0.250000	0.189759	0.021655	0.113861
8	0.216667	0.262048	0.021655	0.150990
9	0.466667	0.358434	0.029389	0.304455
10	0.533333	0.536145	0.029389	0.346535

```
df_normalized.plot(kind='line')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x25ed23b05f8>
```



จัดแบ่งข้อมูลเพื่อนำไป training และ testing data

```
df_test = df_normalized
```

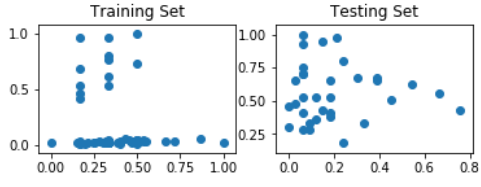
```
df_test
testsize = 32
trainsize = df_test[:~testsize].shape[0]
trainsize
```

```
73
```

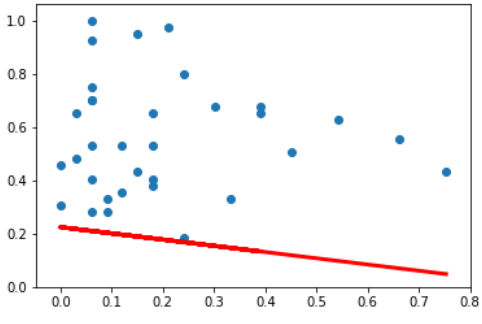
```
X_train = df_test[[0]][:~testsize]
X_test = df_test[[1]][trainsize:]
Y_train = df_test[2][:~testsize]
Y_test = df_test[3][trainsize:]
```

```
plt.subplot(2, 2, 1)
plt.scatter(X_train, Y_train)
plt.title('Training Set')
plt.subplot(2, 2, 2)
plt.scatter(X_test, Y_test)
plt.title('Testing Set')
```

```
Text(0.5,1,'Testing Set')
```




```
model = linear_model.LinearRegression()
model.fit(X_train, Y_train)

Y_pred = model.predict(X_test)

plt.scatter(X_test, Y_test)
plt.plot(X_test, Y_pred, color='red', linewidth = 3)
plt.show()
```



สร้างตัวแบบโดยใช้ค่าประสิทธิภาพ Mean Square Error(MSE)

```
from sklearn import metrics
print('mean_absolute_error = ',metrics.mean_absolute_error(Y_test, Y_pred))
print('mean_squared_error = ',metrics.mean_squared_error(Y_test, Y_pred))
print('mean_squared_error = ',np.sqrt(metrics.mean_squared_error(Y_test, Y_pred)))
```

```
mean_absolute_error =  0.3824083488179291
mean_squared_error =  0.19296742546708792
mean_squared_error =  0.4392805771566595
```

สรุป

การสร้างตัวแบบโดยใช้ค่าประสิทธิภาพ Mean Sqare Error(MSE) ได้ค่า = 0.4392805771566595