Hw 1 Report

Hw 1 - Data Pre-processing

AU7008 Data Mining, SJTU, 2023 Spring

By **Prof. X. He**

Table of Contents

Problem Specification

For the given data (with examples shown below), apply:

- apply *chi-square test of independence* and report the chi-square value, for attributes 性别 w.r.t. 平均睡眠时长;
- apply *0-1 normalization*, for column 薪资;
- apply *0-mean normalization*, for column 薪资.

ID	性别 （1 for male ; 0 for female）	平均睡眠时间 (>6.5小时）	薪资 （元）
1	1	0	13000
2	0	1	13450
3	0	1	11950
…	…	…	…

Result

Chi-Square Independence Test

The parsed table is:

性别	睡眠时间 >6.5h	睡眠时间 <=6.5h	总计
男	12	23	35
女	7	8	15

, from which, we may get the results:

卡方值	p 值	自由度	期望频数
0.6831595116824323	0.7106467871272731	2	[[13.3 21.7 35.] [5.7 9.3 15.]]

0-1 Normalization

The results are:

```
[
    0.318318  , 0.34239247, 0.26214423, 0.40659105, 0.20864541,
    0.10164776, 0.13374706, 0.21399529, 0.204419  , 0.        ,
    0.1910443 , 0.01872459, 0.87203082, 0.19826664, 0.04734646,
    0.08902204, 0.32864327, 0.07607533, 0.05622726, 0.20757543,
    0.39664027, 0.33212069, 0.49229617, 0.31007918, 0.21881019,
    0.26663813, 0.36261502, 0.13759897, 0.09009202, 0.06259362,
    0.13406805, 1.        , 0.27969185, 0.13717098, 0.15273914,
    0.40075968, 0.15659105, 0.14514231, 0.0862401 , 0.28258078,
    0.22533704, 0.39278836, 0.21784721, 0.12582923, 0.07393537,
    0.1771881 , 0.06644554, 0.11619944, 0.09907982, 0.07393537
]
```

0-Mean Normalization

The results are:

```
[
    4.97777912e-01,  6.27304136e-01,  1.95550057e-01,  9.72707399e-01,
    -9.22859951e-02, -6.67958100e-01, -4.95256469e-01, -6.35023899e-02,
    -1.15025043e-01, -1.21484660e+00, -1.86984056e-01, -1.11410398e+00,
    3.47688106e+00, -1.48126189e-01, -9.60111693e-01, -7.35887408e-01,
    5.53330271e-01, -8.05543733e-01, -9.12330909e-01, -9.80427162e-02,
    9.19169893e-01,  5.72039614e-01,  1.43382075e+00,  4.53451160e-01,
    -3.75971452e-02,  2.19728286e-01,  7.36106164e-01, -4.74532273e-01,
    -7.30130687e-01, -8.78078418e-01, -4.93529452e-01,  4.16538489e+00,
     2.89960283e-01, -4.76834961e-01, -3.93074670e-01,  9.41333269e-01,
    -3.72350474e-01, -4.33947389e-01, -7.50854883e-01,  3.05503429e-01,
    -2.48114677e-03,  8.98445697e-01, -4.27781941e-02, -5.37856204e-01,
    -8.17057175e-01, -2.61533594e-01, -8.57354223e-01, -5.89666694e-01,
    -6.81774231e-01, -8.17057175e-01
]
```

Python Implementation

```python
import pandas as pd
import numpy as np
from prettytable import PrettyTable, MARKDOWN
from scipy.stats import chi2_contingency

IN_DATA_PATH = "./data/data.xlsx"

df = pd.read_excel(IN_DATA_PATH, sheet_name=0)

# print(df.head())
# data_sex = df["性别\n（1 for male；0 for female）"]
# data_sleep = df["平均睡眠时间 \n(>6.5小时）"]
# data_salary = df["薪资\n（元）"]
# print(data_sex, data_sleep, data_salary)

data = df.to_numpy()  # (N, 4)
# print(data.shape)

# independent chi^2 value
print(r"Parsing Data & Calculate Independent \chi^2 Value ...")
_ref_male = np.where(1 == data[:, 1])
_ref_female = np.where(0 == data[:, 1])
_cnt_male, _cnt_female = len(_ref_male[0]), len(_ref_female[0])
_sleep_male = np.sum(data[_ref_male][:, 2])
_sleep_female = np.sum(data[_ref_female][:, 2])
chi_data = [
    [_sleep_male, _cnt_male - _sleep_male, _cnt_male],
    [_sleep_female, _cnt_female - _sleep_female, _cnt_female]
]  # [ [male-sleep-gt, male-sleep-gte, male_total], [female-sleep-gt, female-sleep-gte, female_total],]
# parsed data visualization
tbl = PrettyTable()
tbl.set_style(MARKDOWN)
tbl.field_names = ["性别", "睡眠时间 >6.5h", "睡眠时间 <=6.5h", "总计", ]
tbl.add_row(["男", ] + chi_data[0])
tbl.add_row(["女", ] + chi_data[1])
print("===> Parsed")
print(tbl)
res_chi2 = chi2_contingency(chi_data)
print("===> Result")
tbl.clear()
tbl.field_names = ["卡方值", "p 值", "自由度", "期望频数", ]
tbl.add_row(res_chi2)
# tbl.add_row([res_chi2.statistic, res_chi2.pvalue, res_chi2.dof, res_chi2.expected_freq, ])
print(tbl)
print()

# (0,1) Norm + 0-mean Norm
print(r"Applying (0,1) Norm ...")
salary_data = data[:, 3]
_salary_max, _salary_min = np.max(salary_data), np.min(salary_data)
```

```
res_salary_01_norm = (salary_data - _salary_min) * 1. / (_salary_max -
_salary_min)
print("Result ===>\n\t%r" % res_salary_01_norm)
print(r"Applying 0-mean Norm ...")
res_salary_0_std_norm = (salary_data - np.mean(salary_data)) * 1. /
np.std(salary_data)
print("Result ===>\n\t%r" % res_salary_0_std_norm)
```