과제 보고서

(R-Py Computing Homework 1)

과목명	AI+X R-PY 컴퓨팅(AIX0004)
담당 교수님	이정환 교수님
제출일	2019년 10월 28일(월요일)
소속	한양대학교 공과대학 컴퓨터소프트웨어학부
학번	**이름**
2019009261	최가온(CHOI GA ON)

-목차-

Ⅰ. Part 1: 네이버 뉴스 크롤러 만들기 및 검색

다음의 라이브러리를 사용하여 과제를 수행하였다.

```
import urllib.request
from urllib.parse import quote
from bs4 import BeautifulSoup
import re
```

신문사 마다 html tag의 id값이나 class 값이 달라 크롤러를 만들기 어려우나, 네이버 폼으로 바뀐 뉴스에 대해서는 다음과 같은 뚜렷한 url 주소가 있고, html tag의 규칙성이 존재하므로 크롤링(crawling)이 용이하다.

※ 네이버 뉴스에서 '이자율'을 검색어로 하고, 검색 기간을 2019년 10월 13일부터 2019년 10월 14일까지 선택하면 다음과 같은 url을 얻을 수 있다.

https://search.naver.com/search.naver?where=news&query=**이자율**&sm=tab_opt&sort=0&photo=0&field=0&reporter_article=&pd=3&ds=**2019.10.13**&de=**2019.10.14**

1. **문제 1** – Urllib, BeautifulSoup을 이용한 네이버 뉴스 링크 수집

가. urllib.request.urllib를 활용하여 "금리"를 키워드로 동일한 기간의 뉴스페이지를 열고 req란 이름의 객체로 저장한다.

이때, 검색어는 한국어로 표시되기 때문에 별도의 작업이 필요한데, url은 기본적으로 아스키 문자만 넣을 수 있다. 이러한 이유로 한글을 넣기 위해서는 퍼센트 인코딩이 추가적으로 필요하다.

```
# [Q1 - 1]  URL 열기
key_words = urllib.parse.quote("금리")
url = "https://search.naver.com/search.naver?where=news&query=" + key_words + "&sm=tab_opt&sort=0&photo=0&field=0&reporter_article=&pd=3&ds=2019.10.13&de=2019.10.14"
req = urllib.request.urlopen(url)
print("\tOpened an URL Successfully!")
```

나. 이 req 객체에 read() 메소드를 사용하여 data 객체를 만든다.

```
# [Q1 - 2]  read() 메소드 사용
data = req.read().decode('utf8')
```

다. BeautifulSoup을 사용하여 data 객체를 soup이라는 beautifulsoup 함수를 통해

```
# [Q1 - 3]  BeautifulSoup 객체 생성
soup = BeautifulSoup(data, 'html.parser')
```

생성된 객체로 전환시킨다. 옵션은 html.parser를 이용한다.

라. 이 soup 객체에 findAll 메소드를 사용하여 tag값이 a인 것을 모두 수집하여 anchor_set이라는 객체를 만든다.

```
# [Q1 - 4]  findAll() 메소드 사용 및 특정 태그 수집
anchor_set = soup.findAll('a')
```

마. 각 링크의 주소를 저장할 수 있는 news_link란 리스트를 생성한다.

```
# [Q1 - 5]  링크 주소 저장을 위한 리스트 생성
news_link = []
```

바. 이 anchor_set에 for loop를 사용하여, 이 anchor_set에서 하나의 요소들에 대해 만약 href 속성이 'http://news.naver.com/main/read.nhn'으로 시작한다면 news_link란 객체에 이 링크 주소를 받아서 append 메소드를 통해 주소들을 저장한다.

```
# [Q1 - 6]  리스트에 링크 주소 넣기
print("\tGetting articles' links...")
for x in range(len(anchor_set)):
    if re.search('^https://news.naver.com/main/read.nhn',
anchor_set[x]['href']):
        news_link.append(anchor_set[x]['href'])
print("\tArticles' links were completely collected!")
```

2. **문제 2** – '금리'와 관련한 모든 페이지의 뉴스 기사 수집

뉴스 검색과 연관된 모든 기사의 개수를 구한다.

```
# 뉴스 검색과 연관된 총 기사 수 구하기
count_tag = soup.find("div", {"class", "title_desc all_my"})
count_text = count_tag.find("span").get_text().split()
total_num = count_text[-1][0:-1].replace(",", "")
print("\t%d articles were found." % int(total_num))
```

웹 브라우저 상의 한 페이지 단위로 기사를 탐색한다. 이때 URL의 규칙성은 다음과 같다.

```
https://search.naver.com/search.naver?where=news&query=" + key_words +
"&sm=tab_opt&sort=0&photo=0&field=0&reporter_article=&pd=3&ds=2019.10.13&de=20
19.10.14&docid=&nso=so:r,p:from20191013to20191014,a:all&mynews=0&cluster_rank=
26&start=0
```

위를 바탕으로 soup 객체를 생성한 후 URL이 https://news.naver.com/main/read.nhn 으로 시작하는 기사를 news_link 리스트에 append한다. 이때 이 과정은 상당히 오랜 시간이 소요될 가능성이 있어, 사용자 입장에서 전체 과정 중 어느 정도가 수행되었는지를 시각적으로 표시해줄 필요가 있다. 이를 위해 진행도를 수치적/시각적으로 나타내어 줄 수 있는 ProgressBar을 구현하게 되었다. 먼저, news_link 리스트에 URL을 append하는 코드를 보인 후, ProgressBar의 구현 코드를 제시하겠다.

```
print("\tCollecting URLs of articles...\n\tPLEASE WAIT UNTIL THE END>>>")
for val in range(int(total_num) // 10):
    start_val = 10 * val + 1
    if val == 1:
        url2 = "https://search.naver.com/search.naver?where=news&query=" +
key_words +
"&sm=tab_opt&sort=0&photo=0&field=0&reporter_article=&pd=3&ds=2019.10.13&de=2
019.10.14&docid=&nso=so:r,p:from20191013to20191014,a:all&mynews=0&cluster_ran
k=26&start=0" + str(
            start_val) + "&refresh_start=0"
    else:
        url2 = "https://search.naver.com/search.naver?where=news&query=" +
key_words +
"&sm=tab_opt&sort=0&photo=0&field=0&reporter_article=&pd=3&ds=2019.10.13&de=2
019.10.14&docid=&nso=so:r,p:from20191013to20191014,a:all&mynews=0&cluster_ran
k=26&start=" + str(
            start_val) + "&refresh_start=0"
    req2 = urllib.request.urlopen(url2)

    data2 = req2.read().decode('utf8')
    soup2 = BeautifulSoup(data2, 'html.parser')
    anchor_set2 = soup2.findAll('a')

    for x in range(len(anchor_set2)):
        if re.search('^https://news.naver.com/main/read.nhn',
anchor_set2[x]['href']):
            news_link.append(anchor_set2[x]['href'])
    printProgress(val, int(total_num) // 10, prefix='START', suffix='END',
decimals=1, barLength=100)
print("\n\tCompleted!")
```

```
def printProgress(iteration, total, prefix='', suffix='', decimals=1,
barLength=100):
    formatStr = "{0:." + str(decimals) + "f}"
    percent = formatStr.format(100 * (iteration / float(total)))
    filledLength = int(barLength * iteration / float(total))
    bar = '#' * filledLength + '-' * (barLength - filledLength)
    sys.stdout.write('\r%s |%s| %s%s %s' % (prefix, bar, percent, '%',
suffix)),
    if iteration > total:
        sys.stdout.write("\n")
        sys.stdout.flush()
```

3. **문제 3** – 태그를 이용한 뉴스 제목과 뉴스 본문 내용 수집

문제 3을 다루기 전에, 문제 2에서 수집한 기사들은 리스트로 수집하였기 때문에 중복된 기사가 포함되었을 가능성이 있다. 파이썬에서 list 자료형은 여러 자료들을 일련의 순서를 매겨 저장한다. 반면 set은 list와 달리 일련의 순서를 매기지 않고, 그저 포함/미포함의 관계만 나타낼 뿐이다. 이러한 set과 list의 차이점을 이용하여 중복성의 문제를 해결할 수 있다. 기존의 news_link 리스트를 set으로 바꾸면 중복이 사라지고 이를 다시 list로 전환하면 된다.

```
# 중복된 기사 제거
news_link = list(set(news_link))
title_list = []  # 뉴스 제목
text_list = []  # 뉴스 본문 내용
```

중복된 기사를 삭제하면 우리가 이후에 수행할 기사 제목과 본문을 또 다른 리스트를 만들어 append하는 데에 걸리는 시간을 조금이나마 단축시킬 수 있다는 장점이 있다.

```
print("\tCollecting titles and texts of articles...\n\tPLEASE WAIT UNTIL THE
END>>>")
for article in range(len(news_link)):
    news = urllib.request.urlopen(news_link[article])
    soup_news = BeautifulSoup(news, 'html.parser')
    title_list.append(soup_news.find("h3", {"id":
"articleTitle"}).get_text())

    contents = soup_news.find("div", {"id":
"articleBodyContents"}).get_text()
    contents = cleaning_contents(contents)
    text_list.append(contents)
    printProgress(article, len(news_link), prefix='START', suffix='END',
decimals=1, barLength=100)

print("\n\tCompleted!")
```

4. **문제 4** – 정규표현식을 사용한 특정 조건을 만족하는 뉴스 기사 제목 출력

```
# [Q4]  정규표현식을 통한 뉴스 제목 출력 ------------------------------------
print("\n[문제 4 번]")

print("\tTitles of articles which inclues \"금리\" and \"인하\"\n")
a=1
for title in range(len(title_list)):
    if re.search('.*금리.*인하.*', title_list[title]):
        print("\t%2d]  %s" % (a, title_list[title]))
        a+=1
```

5. 전체 코드

```
import urllib.request
from urllib.parse import quote
from bs4 import BeautifulSoup
import re
import time

import sys


def printProgress(iteration, total, prefix='', suffix='', decimals=1,
barLength=100):
    formatStr = "{0:." + str(decimals) + "f}"
    percent = formatStr.format(100 * (iteration / float(total)))
    filledLength = int(barLength * iteration / float(total))
    bar = '#' * filledLength + '-' * (barLength - filledLength)
    sys.stdout.write('\r%s |%s| %s%s %s' % (prefix, bar, percent, '%', suffix)),
    if iteration > total:
        sys.stdout.write("\n")
        sys.stdout.flush()


def cleaning_contents(contents):
    contents = re.sub('\/\/.*\n.*\{\}\n*', '', contents)
    return contents.strip()
    # 기사의 본문 이외의 내용을 모두 지움.
    # strip()을 통해 앞뒤의 공백을 지우는 함수이다.


# [Q1]  링크 수집 ----------------------------------------------------------------
----------------------------------
print("[문제 1번]")
# [Q1 - 1]  URL 열기
key_words = urllib.parse.quote("금리")
url = "https://search.naver.com/search.naver?where=news&query=" + key_words +
"&sm=tab_opt&sort=0&photo=0&field=0&reporter_article=&pd=3&ds=2019.10.13&de=2019
.10.14"
req = urllib.request.urlopen(url)
print("\tOpened an URL Successfully!")

# [Q1 - 2]  read() 메소드 사용
data = req.read().decode('utf8')
# [Q1 - 3]  BeautifulSoup 객체 생성
soup = BeautifulSoup(data, 'html.parser')
# [Q1 - 4]  findAll() 메소드 사용 및 특정 태그 수집
anchor_set = soup.findAll('a')
# [Q1 - 5]  링크 주소 저장을 위한 리스트 생성
news_link = []
# [Q1 - 6]  리스트에 링크 주소 넣기
print("\tGetting articles' links...")
for x in range(len(anchor_set)):
    if re.search('^https://news.naver.com/main/read.nhn',
anchor_set[x]['href']):
        news_link.append(anchor_set[x]['href'])
print("\tArticles' links were completely collected!")
```

```
# [Q2]  특정 검색어가 포함된 기사 찾기 ----------------------------------------
---------------------------------
print("\n[문제 2 번]")
# 뉴스 검색과 연관된 총 기사 수 구하기
count_tag = soup.find("div", {"class", "title_desc all_my"})
count_text = count_tag.find("span").get_text().split()
total_num = count_text[-1][0:-1].replace(",", "")
print("\t%d articles were found." % int(total_num))

print("\tCollecting URLs of articles...\n\tPLEASE WAIT UNTIL THE END>>>")
for val in range(int(total_num) // 10):
    start_val = 10 * val + 1
    if val == 1:
        url2 = "https://search.naver.com/search.naver?where=news&query=" +
key_words +
"&sm=tab_opt&sort=0&photo=0&field=0&reporter_article=&pd=3&ds=2019.10.13&de=2019
.10.14&docid=&nso=so:r,p:from20191013to20191014,a:all&mynews=0&cluster_rank=26&s
tart=0" + str(
            start_val) + "&refresh_start=0"
    else:
        url2 = "https://search.naver.com/search.naver?where=news&query=" +
key_words +
"&sm=tab_opt&sort=0&photo=0&field=0&reporter_article=&pd=3&ds=2019.10.13&de=2019
.10.14&docid=&nso=so:r,p:from20191013to20191014,a:all&mynews=0&cluster_rank=26&s
tart=" + str(
            start_val) + "&refresh_start=0"
    req2 = urllib.request.urlopen(url2)

    data2 = req2.read().decode('utf8')
    soup2 = BeautifulSoup(data2, 'html.parser')
    anchor_set2 = soup2.findAll('a')

    for x in range(len(anchor_set2)):
        if re.search('^https://news.naver.com/main/read.nhn',
anchor_set2[x]['href']):
            news_link.append(anchor_set2[x]['href'])
    printProgress(val, int(total_num) // 10, prefix='START', suffix='END',
decimals=1, barLength=100)
print("\n\tCompleted!")

# [Q3]  뉴스 제목과 본문 수집하기 ----------------------------------------------
---------------------------------
print("\n[문제 3 번]")
# 중복된 기사 제거
news_link = list(set(news_link))
title_list = []  # 뉴스 제목
text_list = []  # 뉴스 본문 내용

print("\tCollecting titles and texts of articles...\n\tPLEASE WAIT UNTIL THE
END>>>")
for article in range(len(news_link)):
    news = urllib.request.urlopen(news_link[article])
    soup_news = BeautifulSoup(news, 'html.parser')
    title_list.append(soup_news.find("h3", {"id": "articleTitle"}).get_text())
```

```
    contents = soup_news.find("div", {"id": "articleBodyContents"}).get_text()
    contents = cleaning_contents(contents)
    text_list.append(contents)
    printProgress(article, len(news_link), prefix='START', suffix='END',
decimals=1, barLength=100)

print("\n\tCompleted!")

# [Q4]  정규표현식을 통한 뉴스 제목 출력 ----------------------------------------
---------------------------------
print("\n[문제 4번]")

print("\tTitles of articles which inclues \"금리\" and \"인하\"\n")
a=1
for title in range(len(title_list)):
    if re.search('.*금리.*인하.*', title_list[title]):
        print("\t%2d]   %s" % (a, title_list[title]))
        a+=1
```

(실행결과)

```
C:₩Users₩choig₩Anaconda3₩envs₩pytorch₩python.exe C:/Users/choig/PycharmProjects/끄쩎끄쩎/HAI.py
[문제 1번]
        Opened an URL Successfully!
        Getting articles' links...
        Articles' links were completely collected!

[문제 2번]
        1198 articles were found.
        Collecting URLs of articles...
        PLEASE WAIT UNTIL THE END>>>
START |#################################################################################################################-| 99.2% END
        Completed!

[문제 3번]
        Collecting titles and texts of articles...
        PLEASE WAIT UNTIL THE END>>>
START |#################################################################################################################-| 99.8% END
        Completed!

[문제 4번]
        Titles of articles which inclues "금리" and "인하"

         1]   잇따른 'D 논란' 급해진 한국은행, 오는 16일 기준금리 인하로 기운다
         2]   채권 전문가 10명 중 8명은 "한은 금리인하"
         3]   '사상 최저금리' 향하는 한은…16일 금리인하 나서나
         4]   [금통위 풀] 전문가 100% "10월에 기준금리 인하"
         5]   마이너스 물가·경기부진에…한은, 이번주 금리인하 할듯
         6]   [마켓 Watch] "韓銀 16일 금리인하 확률 90%"
         7]   10월 금리인하 무게…추가 인하 '촉각'
         8]   경기둔화에 디플레 우려까지…"한은 이번주 금리인하할듯"
         9]   [경제전망]한은, 금리 인하 나설까…9월 취업자수 증가 '주목'
        10]   한은 이번 주 기준금리 인하 '유력'
```

```
11]  [금통위 폴] "기준금리, 내년 상반기 1%까지 추가 인하"
12]  금리인하 기대 속 9월 외국인 채권자금 11억 달러 순유입
13]  [금통위 폴] "이주열, 신호 충분히 줘"...10월 금리인하 전망 100%
14]  보험연구원 "LAT 변화·금리 인하...금리 추가인하시 보험사 부담↑"
15]  [오늘의 키워드] "한은, 금주 금리인하할 듯...이주열 발언 시그널로 충분"
16]  3분기 실적전망 개선·금리인하...훈풍 부는 코스피
17]  한은 16일 기준금리 인하 가능성 ↑...경기 회복세 지원에 초점
18]  [마감시황] 코스피, 미중 휴전·금리 인하 가능성에 2060선 회복
19]  높아진 10월 금리 인하 가능성...고민되는 부동산 가격 상승
20]  경기둔화에 디플레이션 우려까지..."한은 이번 주 금리인하할듯"
21]  금리인하,양적완화 놓고 '내전' 휩싸인 美^유럽 중앙은행
22]  [fn논단]금리 인하냐 동결이냐 그것이 문제로다
23]  금리인하 기대 속 9월 외국인 채권자금 11억달러 순유입
24]  경기둔화에 디플레 우려까지..."한은 이번주 금리인하할 듯"
25]  한은 금통위원들 "물가 낮아 고민...금리인하 여력 남아있다"
26]  경기둔화에 한은, 이번주 금리인하하나...금통위원들 "디플레 때 양적완화 고려...지금은 아냐"

Process finished with exit code 0
```

Ⅱ. 보스턴 주택가격 데이터 분석하기

다음의 라이브러리를 사용하여 과제를 수행하였다.

```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
```

1. **문제 5** – 데이터 전처리 시행

가. 제공된 boston_csv.csv 파일을 사용하여 pandas 데이터 프레임 객체를 만든다. 결측치 코드인 na와 NaN이 모두 실제 결측치로 되도록 한다.

```
# [Q5 - 1]  pandas 데이터 프레임 객체 생성
path = "C:\\Users\\choig\\Desktop\\파일\\1 학년 2 학기\\AIX R-PY
컴퓨팅\\과제\\Part 2\\boston_csv.csv"
df = pd.read_csv(path)
print("Successfully opened csv file.")

# Data Cleaning
df = df.replace("na", np.nan)       # "na" --> np.nan
df = df.replace("Nan", np.nan)      # "Nan" --> np.nan
print("\"na\", \"Nan\" --> np.nan")
```

나. 이와 같은 결측치가 있는 관측치를 모두 제거한다.

```
# [Q5 - 2]  결측치를 포함한 관측치 삭제
df = df.dropna(axis=0)  # 해당 열을 삭제
print("Performed Data Cleaning successfully.")
```

2. **문제 6** – 요약 통계 구하기

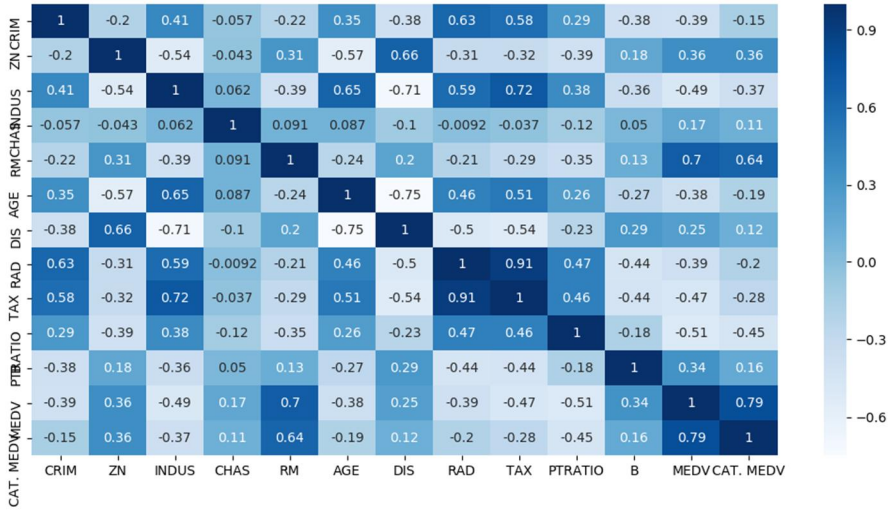
가. describe 메소드를 적용해서 각 변수명 요약 통계를 구한다.

```
# [Q6 - 1]  변수별 요약 통계
print(df.describe())
```

나. 상관관계를 구한 후 seaborn 라이브러리의 heatmap을 구현한다.

위의 heatmap를 표시하면 다음과 같은 결과를 얻을 수 있다.

```
# [Q6 - 2]  Heatmap 구현
plt.figure(figsize=(15, 15))
sns.heatmap(data=df.corr(), annot=True, cmap='Blues')
plt.show()  # Plot 표시
```



3. **문제 7** – 단순회귀분석 모형

독립변수와 종속변수를 설정한다. 독립변수는 모집단의 하위계층의 비율인 LSTAT이고, 종속변수는 본인 소유의 주택가격(중앙값)인 MEDV이다.

```
x1 = df[['LSTAT']]  # 독립변수: 하위계층의 비율(LSTAT)
y1 = df['MEDV']     # 종속변수: 본인 소유의 주택가격(중앙값)(MEDV)
```

Training set이 표본의 75%를 차지한다. 즉, 표본 분할(Sample Split)을 하여 일부 데이터는 Training set으로, 나머지 데이터는 예측력을 확인하는 목적으로 하는 Testing set으로 나눈다는 것이다.

```
x1_train, x1_test, y1_train, y1_test = train_test_split(x1, y1, test_size=1 - 0.75, random_state=0)
# Training Set: x1_train, y1_train // Test Set: x1_test, y1_test
# Training Set 이 표본의 75%를 차지한다.
```

LinearRegression 객체를 생성한 후, 단순 회귀 모형을 구현하기 위해 fitting 작업과 predicting을 한다.

```
lm = LinearRegression()
lm.fit(x1_train, y1_train)
y_hat = lm.predict(x1_train)
```

Training set에 대해서 회귀 분석 추정 계수, R^2 값 그리고 mean squared error 값을 산출하면 다음과 같다.

이때, 회귀 분석 추정 계수는 lm.coef_와 lm.intercept_로 구할 수 있다. 또한 R^2 값은 lm.score()를 통해, 그리고 mean squared error 값은 mean_squared_error()를 통해 구할 수 있다.

```
# 상관관계 분석
print("MEDV", " = ", lm.coef_[0], " * ", "LSTAT", " + ", lm.intercept_,
sep="")
print(lm.coef_, lm.intercept_)
print("\nThe model performance for train set")
# Mean Squared Error
print("\tmean squared error: ", mean_squared_error(y1_train, y_hat))
# R Square
print("\tR square: ", lm.score(x1_train, y1_train))
```

위에서 LinearRegression 객체인 lm에 x1_train와 y1_train을 바탕으로 Fitting을 하였다는 사실을 상기하라. 이렇게 Linear Regression된 상황을 바탕으로 x1_test와 y1_test를 통해 제대로 모델이 예측력을 가지고 있는지를 시험할 것이다.

```
print("\nThe model performance for test set")
y_pred1 = lm.predict(x1_test)
# Mean Squared Error
print("\tmean squared error: ", mean_squared_error(y1_test, y_pred1))
```

4. **문제 8** – 다중회귀분석 모형

독립변수와 종속변수를 설정한다. 독립변수는 모집단의 하위계층의 비율인 LSTAT과 10,000 달러 당 재산세율인 TAX이며, 종속변수는 본인 소유의 주택가격(중앙값)인 MEDV이다.

```
x2 = df[['LSTAT', 'TAX']] # 독립변수: 하위계층의 비율(LSTAT), 10,000 달러 당
재산세율(TAX)
y2 = df['MEDV'] # 종속변수: 본인 소유의 주택가격(중앙값)(MEDV)
```

문제 7에서와 마찬가지로 Sample Split을 시행한다. 위와 동일하게 Training Set이 전체 표본의 75%를 차지한다.

```
x2_train, x2_test, y2_train, y2_test = train_test_split(x2, y2, test_size=1 -
0.75, random_state=0)
# Training Set: x2_train, y2_train // Test Set: x2_test, y2_test
# Training Set 이 표본의 75%를 차지한다.
```

LinearRegression 객체를 생성한 후, 단순 회귀 모형을 구현하기 위해 fitting 작업과 predicting을 한다. 이때 문제 7에서 사용한 LinearRegression 객체와 구분하기 위해 객체의 이름을 lm2로 하였다.

```
lm2 = LinearRegression()
lm2.fit(x2_train, y2_train)
y_hat2 = lm2.predict(x2_train)
```

Training set에 대해서 회귀 분석 추정 계수, R^2 값 그리고 mean squared error 값을 산출하면 다음과 같다.

```
# 상관관계 분석
print("MEDV", " = ", lm2.coef_[0], " * ", "LSTAT", " + ", lm2.coef_[1], " *
", "TAX", " + ", lm2.intercept_, sep="")
print(lm2.coef_, lm2.intercept_)
print("\nThe model performance for train set")
# Mean Squared Error
print("\tmean squared error: ", mean_squared_error(y2_train, y_hat2))
# R Square
print("\tR square: ", lm2.score(x2_train, y2_train))
```

문제 7에서와 마찬가지로 lm2 객체의 Fitting은 선행되었다. 이를 바탕으로 test set을 통해 모델의 예측력을 평가해보자.

```
y_pred2 = lm2.predict(x2_test)
# Mean Squared Error
print("\tmean squared error: ", mean_squared_error(y2_test, y_pred2))
```

5. 전체 코드

여기에 「Part 2: 보스턴 주택가격 데이터 분석하기」에 대해 작성된 코드의 전체 부분을 첨부한다. 또한 그 코드를 실행했을 때의 결과를 함께 보인다.

```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns

# [Q5]  데이터 전처리 ------------------------------------------------------------
-------------------------------------
print("[문제 5번]\n")
# [Q5 - 1]  pandas 데이터 프레임 객체 생성
path = "C:\\Users\\choig\\Desktop\\파일\\1학년 2학기\\AIX R-PY 컴퓨팅\\과제\\Part
2\\boston_csv.csv"
df = pd.read_csv(path)
```

```
print("Successfully opened csv file.")

# Data Cleaning
df = df.replace("na", np.nan)       # "na" --> np.nan
df = df.replace("Nan", np.nan)      # "Nan" --> np.nan
print("\"na\", \"Nan\" --> np.nan")
# [Q5 - 2]  결측치를 포함한 관측치 삭제
df = df.dropna(axis=0)  # 해당 열을 삭제
print("Performed Data Cleaning successfully.")
# ------------------------------------------------------------------------------
-------------------------------------

# [Q6]  요약 통계 ---------------------------------------------------------------
-----------------------------------
print("\n\n[문제 6번]\n")
# [Q6 - 1]  변수별 요약 통계
print(df.describe())
# [Q6 - 2]  Heatmap 구현
plt.figure(figsize=(15, 15))
sns.heatmap(data=df.corr(), annot=True, cmap='Blues')
plt.show()  # Plot 표시
# ------------------------------------------------------------------------------
-------------------------------------

# [Q7]  단순회귀분석 모형 -------------------------------------------------------
-----------------------------------
print("\n\n[문제 7번]\n")
x1 = df[['LSTAT']]  # 독립변수: 하위계층의 비율(LSTAT)
y1 = df['MEDV']     # 종속변수: 본인 소유의 주택가격(중앙값)(MEDV)

x1_train, x1_test, y1_train, y1_test = train_test_split(x1, y1, test_size=1 -
0.75, random_state=0)
# Training Set: x1_train, y1_train // Test Set: x1_test, y1_test
# Training Set이 표본의 75%를 차지한다.
lm = LinearRegression()
lm.fit(x1_train, y1_train)
y_hat = lm.predict(x1_train)

# 상관관계 분석
print("MEDV", " = ", lm.coef_[0], " * ", "LSTAT", " + ", lm.intercept_, sep="")
print(lm.coef_, lm.intercept_)
print("\nThe model performance for train set")
# Mean Squared Error
print("\tmean squared error: ", mean_squared_error(y1_train, y_hat))
# R Square
print("\tR square: ", lm.score(x1_train, y1_train))

print("\nThe model performance for test set")
y_pred1 = lm.predict(x1_test)
# Mean Squared Error
print("\tmean squared error: ", mean_squared_error(y1_test, y_pred1))

# [Q8]  다중회귀분석 모형 -------------------------------------------------------
-----------------------------------
print("\n\n[문제 8번]\n")
```

```
x2 = df[['LSTAT', 'TAX']]  # 독립변수: 하위계층의 비율(LSTAT), 10,000 달러 당
재산세율(TAX)
y2 = df['MEDV']  # 종속변수: 본인 소유의 주택가격(중앙값)(MEDV)

x2_train, x2_test, y2_train, y2_test = train_test_split(x2, y2, test_size=1 -
0.75, random_state=0)
# Training Set: x2_train, y2_train // Test Set: x2_test, y2_test
# Training Set이 표본의 75%를 차지한다.
lm2 = LinearRegression()
lm2.fit(x2_train, y2_train)
y_hat2 = lm2.predict(x2_train)

# 상관관계 분석
print("MEDV", " = ", lm2.coef_[0], " * ", "LSTAT", " + ", lm2.coef_[1], " * ",
"TAX", " + ", lm2.intercept_, sep="")
print(lm2.coef_, lm2.intercept_)
print("\nThe model performance for train set")
# Mean Squared Error
print("\tmean squared error: ", mean_squared_error(y2_train, y_hat2))
# R Square
print("\tR square: ", lm2.score(x2_train, y2_train))

print("\nThe model performance for test set")
y_pred2 = lm2.predict(x2_test)
# Mean Squared Error
print("\tmean squared error: ", mean_squared_error(y2_test, y_pred2))
# ------------------------------------------------------------------------------
-------------------------------------
```

(실행결과)

```
C:₩Users₩choig₩Anaconda3₩envs₩pytorch₩python.exe C:/Users/choig/PycharmProjects/끄쩎끄쩎/HAI.py
[문제 5번]

Successfully opened csv file.
"na", "Nan" --> np.nan
Performed Data Cleaning successfully.


[문제 6번]

              CRIM          ZN       INDUS  ...           B       MEDV    CAT. MEDV
count   502.000000  502.000000  502.000000  ...  502.000000  502.000000  502.000000
mean      3.641708   11.418327   11.163765  ...  356.353506   22.564343    0.167331
std       8.629979   23.396912    6.873538  ...   91.587527    9.217580    0.373643
min       0.009060    0.000000    0.460000  ...    0.320000    5.000000    0.000000
25%       0.082492    0.000000    5.190000  ...  375.240000   17.100000    0.000000
50%       0.262660    0.000000    9.690000  ...  391.340000   21.200000    0.000000
75%       3.689387   12.500000   18.100000  ...  396.120000   25.000000    0.000000
```

```
max        88.976200   100.000000     27.740000   ...   396.900000     50.000000       1.000000

[8 rows x 13 columns]


[문제 7번]

MEDV = -0.9788666918716608 * LSTAT + 34.96140049096505
[-0.97886669] 34.96140049096505

The model performance for train set
        mean squared error:   40.27627752592205
        R square:   0.5411319824508232

The model performance for test set
        mean squared error:   32.07645208363777


[문제 8번]

MEDV = -0.9053643302180053 * LSTAT + -0.005554065186702708 * TAX + 36.31384685772977
[-0.90536433 -0.00555407] 36.31384685772977

The model performance for train set
        mean squared error:   39.65545131415304
        R square:   0.5482050614575329

The model performance for test set
        mean squared error:   31.56493746549088

Process finished with exit code 0
```

- 중간에 Heatmap이 출력되는데 그것에 대해서는 위에서 그림으로 첨부하였다.