R – Py 컴퓨팅: Homework 1

학과 : 경영학부

학번 : 2018027383

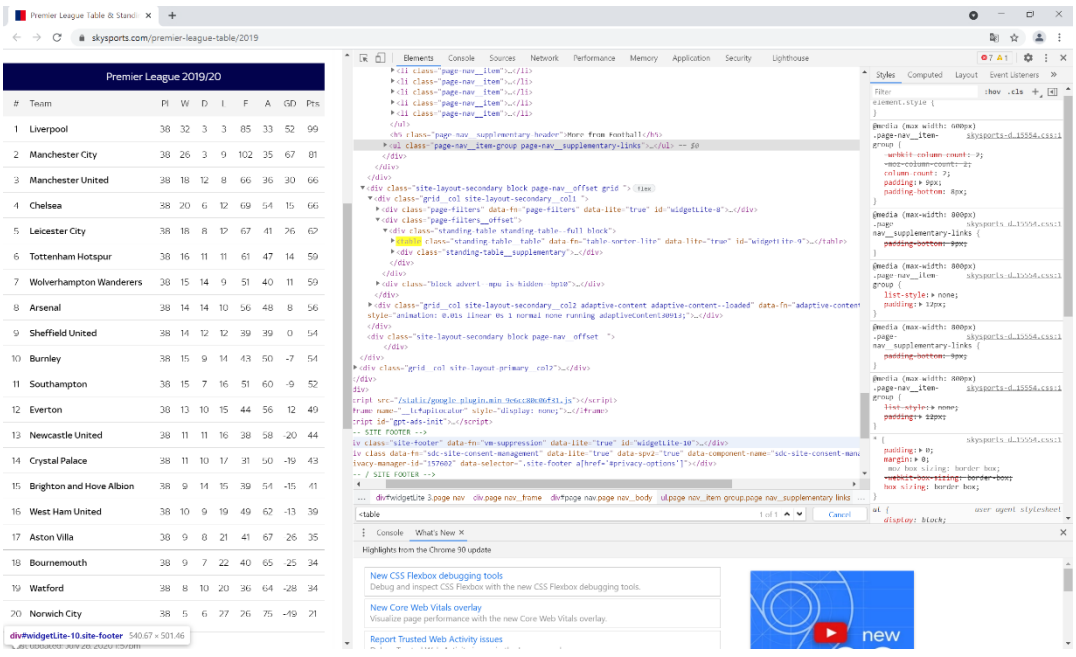
이름 : 손정범

Part 1 영국 축구리그 순위 데이터 크롤링

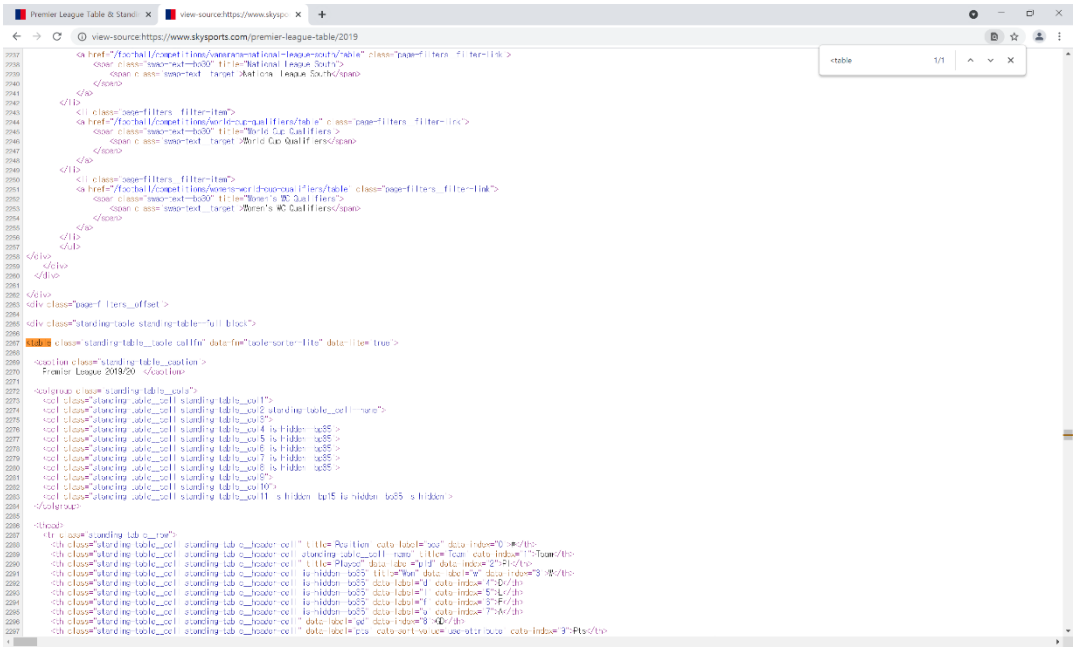
- https://www.skysports.com/premier-league-table/2019 싸이트를 방문하여 본다.
- 여기는 2019-2020년 영국 축구리그인 프리미어리그 20개 팀의 최종성적이 표로 제시되어 있으며 여기서 Pl: 총 게임수 W: 승 D: 무승부, L: 패배 F: 득점 A: 실정 GD:득실차를 나타낸다.
- Part 1과제는 이 20개팀의 최종 성적 테이블을 크롤링하여 데이터 프레임을 만들고 간단한 회귀 분석을 해보는 것이다.

Q1.

A) F12를 사용하여 개발자 도구를 열고 가장 먼저 발견되는 table tag (<table>)를 찾아보고 스크린 샷을 해서 제출하라 (하기 그림 참조 속성값들이 있기 때문에 <table 로 검색하라)



B). 브라우저(크롬/엣지) view-source: https://www.skysports.com/premier-league-table/2019를 하고 ctrl+F (윈도우 기준) 하여 <table을 찾아서 하기와 같이 스크린샷 하라.



C) 이 웹페이지에서 <table> 태그를 가진 html 객체는 하나이다. Q1과 Q2에서 찾은 table tag에 연관된 클래스 속성의 이름은 동일한가 다른가?

A번에서 찾은 table tag에 연관된 클래스 속성의 이름은 standing-table__table이었다. 반면 B번에서 찾은 table tag와 연관된 클래스 속성의 이름은 standing-table__table callfn이었다. 즉 두 개의 클래스 속성은 동일하지 않고 다르다고 볼 수 있다.

Q2 Viewsource 클래스 이름을 기준으로 테이블을 웹크롤링하여 데이터 프레임을 만든다

- **테이블을 웹크롤링하는 방법은 실시간 강의를 대체한 온라인 강의 6주차에 비교적 상세히 설명되어 있으니 참고한다 (아래 스크린샷 참조).**
- **STEP1) 먼저 웹페이지를 urllib나 requests library를 사용하여 열어서 객체로 만든다.**
- **STEP2) BeautifulSoup을 사용하여 이파일을 읽고 soup이라는 객체를 만든다.**
- **STEP3) 이 soup객체에 find (soup.find) 메소드를 적용하여 tag 이름은 "table"이고 class속성은 " standing-table__table callfn"을 찾아 table 객체로 저장한다.**
- **STEP4) headers란 리스트를 만들어서 이 table객체에서 th 태그를 가진 것들의 내용들을 다 모은다. 즉 Pl, W, D, L, …. GD등 여러분이 본 최종순위 테이블의 헤더들을 다 수집한다.**
- **STEP5) df = pd.DataFrame(columns = headers)를 적용하여 데이터 프레임을 만든다.**
- **STEP6) 이 df 에 대해 상위 순위부터 하위순위까지 순서로 데이터 내용을 저장한다.하기 켑쳐본의 코드를 활용하되 strip함수를 사용하여 /n 등 공백 문자열들을 제거한다.**

아래 사진과 같은 데이터프레임을 구성하였다.

```
$ python 1번 .py
     #                    Team  Pl   W   D   L    F   A   GD Pts Last 6
0    1               Liverpool  38  32   3   3   85  33   52  99
1    2         Manchester City  38  26   3   9  102  35   67  81
2    3       Manchester United  38  18  12   8   66  36   30  66
3    4                 Chelsea  38  20   6  12   69  54   15  66
4    5          Leicester City  38  18   8  12   67  41   26  62
5    6       Tottenham Hotspur  38  16  11  11   61  47   14  59
6    7  Wolverhampton Wanderers  38  15  14   9   51  40   11  59
7    8                 Arsenal  38  14  14  10   56  48    8  56
8    9         Sheffield United  38  14  12  12   39  39    0  54
9   10                 Burnley  38  15   9  14   43  50   -7  54
10  11             Southampton  38  15   7  16   51  60   -9  52
11  12                 Everton  38  13  10  15   44  56  -12  49
12  13         Newcastle United  38  11  11  16   38  58  -20  44
13  14           Crystal Palace  38  11  10  17   31  50  -19  43
14  15  Brighton and Hove Albion  38   9  14  15   39  54  -15  41
15  16          West Ham United  38  10   9  19   49  62  -13  39
16  17              Aston Villa  38   9   8  21   41  67  -26  35
17  18              Bournemouth  38   9   7  22   40  65  -25  34
18  19                  Watford  38   8  10  20   36  64  -28  34
19  20            Norwich City  38   5   6  27   26  75  -49  21
```

Q3 승률과 골득실차와의 관계를 회귀 분석을 통해 분석해보고자 한다.

- **STEP1) df에 승률(win_prob)를 승리회수(W)을 전체 게임수(PL)로 나눈 것으로 정의한다. 예를 들어 리버플의 승률=32/38이다. 데이터 프레임의 모든 칼럼이 문자열 형으로 지정되었을 수 있음으로 astype을 활용하여 필요 칼럼을 실수형으로 변환한다. (as type에 관해서는 https://hogni.tistory.com/70 참고)**
- **STEP2) 다음과 같은 library를 import해서 선형회귀 분석을 준비한다.**

  from sklearn.linear_model import LinearRegression

  import numpy as np

- **STEP3) X=df["GD"], Y=df[“win_prob”]하여 Y=a+bX라는 회귀분석을 수행해 보고 a값고 b값을 출력해 본다. 골득실과 승리확률은 어떠한 관계가 있는가 자신의 의견을 기술해보라.**

  a값, 즉 y절편은 0.3789473684210526, b값, 즉 기울기는 0.00550239으로 출력된다. 골득실과 승리확률 간의 관계를 파악하기에 앞서 회귀분석 모델의 설명력을 파악해보고자 결정계수 값을 출력했다. 결정계수는 0.8717574809124105로 나오므로 설명력이 어느정도 있는 편이라고 볼 수 있다. 위 방정식에서 b값은 coef_변수, 다시 말해 각 속성들의 상관계수 값에 해당하는데 양수이므로 양의 영향력을 가지고 있다. 즉 골득실이 클수록 승리확률이 높다고 할 수 있다. 하지만 숫자의 절대값이 크지 않으므로 매우 약한 선형관계를 나타내고 있다고 볼 수 있을 것이다.

Q4 선형회귀 분석의 결과에 대해 p-값과 같은 다양한 통계량을 출력해보고자 한다. 내용을 모른다면 다음 명령어만 수행해보아도 상관없다.

import statsmodels.api as sm

X2 = sm.add_constant(X)

est = sm.OLS(Y, X2)

est2 = est.fit()

print(est2.summary())

- **Q3과 Q4의 절편과 기울기 결과값들은 동일한가?**

  Q3와 Q4에서 구한 절편과 기울기 값은 다음과 같다,

  Q3절편 : 0.3789473684210526 / Q3기울기 : 0.00550239

  Q4절편 : 0.3789 / Q4기울기 : 0.0055

  Q4에서 구한 다양한 통계량은 반올림한 결과값이 출력된 것이므로 두 문제에서 구한 절편과 기울기 결과값은 동일하다고 볼 수 있다.

Part 2 보스턴 주택가격 데이터 분석하기: boston_csv.csv 파일

- **보스턴 주택 가격 데이터의 변수명은 다음과 같다.**

CRIM	자치시(town) 별 1 인당 범죄율
ZN	25,000 평방피트를 초과하는 거주지역의 비율
INDUS	비소매상업지역이 점유하고 있는 토지의 비율
CHAS	찰스강에 대한 더미변수 (강의 경계에 위치한 경우는 1, 아니면 0)
NOX	10ppm 당 농축 일산화질소
RM	주택 1 가구당 평균 방의 개수
AGE	1940 년 이전에 건축된 소유주택의 비율
DIS	5 개의 보스턴 직업센터까지의 접근성 지수
RAD	방사형 도로까지의 접근성 지수
TAX	10,000 달러 당 재산세율
PTRATIO	자치시(town)별 학생/교사 비율
B	1000(Bk-0.63)^2, 여기서 Bk 는 자치시별 흑인의 비율을 말함.
LSTAT	모집단의 하위계층의 비율(%)
MEDV	본인 소유의 주택가격(중앙값) (단위: $1,000)
CAT.MEDV	MEDV 가 $30,000 을 넘는지에 대한 변수 (넘는 경우 1, 아닌 경우 0)

- **보스턴 주택 가격 데이터에서 결측치는 na와 NaN으로 표시되어 있다.**
- **다음과 같은 라이브러리를 불러와야 프로그램이 구현될 것이다.**

from sklearn.linear_model import LinearRegression

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 다음과 같은 데이터 전처리를 시행하라.

1. **제공된 boston_csv.csv 파일을 사용하여 pandas 데이터 프레임 객체를 만든다, 결측치 코드인 na와 NaN이 모두 실제 결측치로 되도록 한다.**
2. **이와 같은 결측치가 있는 관측치를 모두 제거한다.**

```
$ python 2번.py
        CRIM    ZN  INDUS  CHAS    NOX     RM   AGE     DIS  RAD  TAX  PTRATIO       B  LSTAT  MEDV  CAT. MEDV
2    0.02729   0.0   7.07     0  0.469  7.185  61.1  4.9671    2  242     17.8  392.83   4.03  34.7          1
3    0.03237   0.0   2.18     0  0.458  6.998  45.8  6.0622    3  222     18.7  394.63   2.94  33.4          1
4    0.06905   0.0   2.18     0  0.458  7.147  54.2  6.0622    3  222     18.7  396.90   5.33  36.2          1
5    0.02985   0.0   2.18     0  0.458  6.430  58.7  6.0622    3  222     18.7  394.12   5.21  28.7          0
6    0.08829  12.5   7.87     0  0.524  6.012  66.6  5.5605    5  311     15.2  395.60  12.43  22.9          0
..       ...   ...    ...   ...    ...    ...   ...     ...  ...  ...      ...     ...    ...   ...        ...
499  0.17783   0.0   9.69     0  0.585  5.569  73.5  2.3999    6  391     19.2  395.77  15.10  17.5          0
501  0.06263   0.0  11.93     0  0.573  6.593  69.1  2.4786    1  273     21.0  391.99   9.67  22.4          0
502  0.04527   0.0  11.93     0  0.573  6.120  76.7  2.2875    1  273     21.0  396.90   9.08  20.6          0
503  0.06076   0.0  11.93     0  0.573  6.976  91.0  2.1675    1  273     21.0  396.90   5.64  23.9          0
504  0.10959   0.0  11.93     0  0.573  6.794  89.3  2.3889    1  273     21.0  393.45   6.48  22.0          0

[502 rows x 15 columns]
```

Q6 다음과 같은 요약 통계를 구하라.

1. **describe 메소드를 적용해서 각 변수병 요약 통계를 구한다.**

요약통계량

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV	CAT. MEDV
count	502.000000	502.000000	502.000000	502.000000	502.000000	502.000000	502.000000	502.000000	502.000000	502.000000	502.000000	502.000000	502.000000	502.000000	502.000000
mean	3.641708	11.418327	11.163765	0.069721	0.554802	6.284805	68.514542	3.797274	9.605578	409.095618	18.456574	356.353506	12.681514	22.564343	0.167331
std	8.629979	23.396912	6.873538	0.254930	0.116263	0.705085	28.247125	2.111828	8.717100	168.859125	2.165559	91.587527	7.155966	9.217580	0.373643
min	0.009060	0.000000	0.460000	0.000000	0.385000	3.561000	2.900000	1.129600	1.000000	187.000000	12.600000	0.320000	1.730000	5.000000	0.000000
25%	0.082492	0.000000	5.190000	0.000000	0.449000	5.884250	44.550000	2.091150	4.000000	279.250000	17.400000	375.240000	6.950000	17.100000	0.000000
50%	0.262660	0.000000	9.690000	0.000000	0.538000	6.208500	77.150000	3.207450	5.000000	330.000000	19.050000	391.340000	11.395000	21.200000	0.000000
75%	3.689388	12.500000	18.100000	0.000000	0.624000	6.628000	94.100000	5.213925	24.000000	666.000000	20.200000	396.120000	17.057500	25.000000	0.000000
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	100.000000	12.126500	24.000000	711.000000	22.000000	396.900000	37.970000	50.000000	1.000000

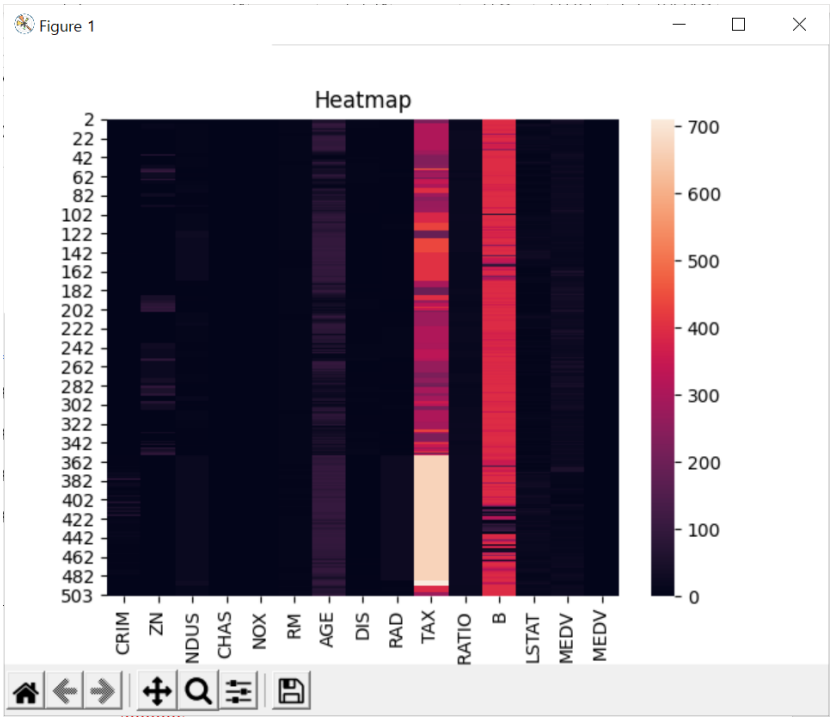
2. **상관관계를 구한후 seaborn 라이브러리의 heatmap 을 구현한다.**

상관관계

상관관계

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV	CAT. MEDV
CRIM	1.000000	-0.201718	0.406051	-0.056841	0.421132	-0.219579	0.354022	-0.380747	0.625027	0.582237	0.290985	-0.384175	0.455111	-0.390650	-0.153676
ZN	-0.201718	1.000000	-0.535297	-0.043384	-0.517580	0.311633	-0.569038	0.664917	-0.314729	-0.316928	-0.391435	0.176832	-0.414400	0.359445	0.364826
INDUS	0.406051	-0.535297	1.000000	0.062010	0.764556	-0.391454	0.646623	-0.709355	0.594547	0.720994	0.381237	-0.356247	0.603096	-0.485291	-0.369083
CHAS	-0.056841	-0.043384	0.062010	1.000000	0.091037	0.091255	0.087172	-0.099601	-0.009157	-0.037064	-0.122132	0.049800	-0.055137	0.174682	0.107779
NOX	0.421132	-0.517580	0.764556	0.091037	1.000000	-0.301822	0.732474	-0.769195	0.612544	0.668506	0.188344	-0.380186	0.591440	-0.427771	-0.233252
RM	-0.219579	0.311633	-0.391454	0.091255	-0.301822	1.000000	-0.239939	0.204346	-0.210409	-0.292516	-0.354572	0.128335	-0.614499	0.695529	0.641968
AGE	0.354022	-0.569038	0.646623	0.087172	0.732474	-0.239939	1.000000	-0.748412	0.459126	0.509096	0.261201	-0.274829	0.604278	-0.376197	-0.190483
DIS	-0.380747	0.664917	-0.709355	-0.099601	-0.769195	0.204346	-0.748412	1.000000	-0.497000	-0.535945	-0.230986	0.292520	-0.498160	0.248532	0.118635
RAD	0.625027	-0.314729	0.594547	-0.009157	0.612544	-0.210409	0.459126	-0.497000	1.000000	0.910025	0.466991	-0.443167	0.486908	-0.386034	-0.201537
TAX	0.582237	-0.316928	0.720994	-0.037064	0.668506	-0.292516	0.509096	-0.535945	0.910025	1.000000	0.462933	-0.440811	0.542725	-0.472305	-0.276752
PTRATIO	0.290985	-0.391435	0.381237	-0.122132	0.188344	-0.354572	0.261201	-0.230986	0.466991	0.462933	1.000000	-0.177965	0.374125	-0.507027	-0.445634
B	-0.384175	0.176832	-0.356247	0.049800	-0.380186	0.128335	-0.274829	0.292520	-0.443167	-0.440811	-0.177965	1.000000	-0.365301	0.335823	0.156951
LSTAT	0.455111	-0.414400	0.603096	-0.055137	0.591440	-0.614499	0.604278	-0.498160	0.486908	0.542725	0.374125	-0.365301	1.000000	-0.741372	-0.472958
MEDV	-0.390650	0.359445	-0.485291	0.174682	-0.427771	0.695529	-0.376197	0.248532	-0.386034	-0.472305	-0.507027	0.335823	-0.741372	1.000000	0.790267
CAT. MEDV	-0.153676	0.364826	-0.369083	0.107779	-0.233252	0.641968	-0.190483	0.118635	-0.201537	-0.276752	-0.445634	0.156951	-0.472958	0.790267	1.000000

Seaborn 라이브러리의 heatmap 구현



Q7 다음과 같은 단순회귀분석 모형을 Training Set과 Test Set을 통해 구현하라.

- 모집단의 하위계층의 비율(LSTAT)이 독립변수
- 본인 소유의 주택가격(중앙값)인 MEDV가 종속변수
- Training set이 표본의 75%를 차지한다.
- Training Set에대해서는 회귀 분석 추정 계수 및 R^2 값 mean squared error 값을 보고한다.

  y절편 : 34.77753065315754

  기울기 : -0.96986967

  결정계수 : 0.5586174894827725

  Mean squared error : 36.81870687251752

- Training Set에대해서는 회귀 분석 추정 계수 값을 바탕으로 Test Set에서 예측한 후 mean squared error 값을 보고한다.

  Mean squared error : 42.35171643712479

Q8 다음과 같은 다중회귀분석 모형을 Training Set과 Test Set을 통해 구현하라.

- 모집단의 하위계층의 비율(LSTAT)과 10,000 달러 당 재산세율(TAX)가 독립변수
- 본인 소유의 주택가격(중앙값)인 MEDV가 종속변수
- Training set이 표본의 75%를 차지한다.
- Training Set에대해서는 회귀 분석 추정 계수 및 R^2 값 mean squared error 값을 보고한다.

y절편 : 36.50257255632521

기울기 : ["LSTAT"] : -0.89452441/ ["TAX"] : -0.00593868

결정계수 : 0.5725225505094894

Mean squared error : 37.41764442060701

- Training Set에대해서는 회귀 분석 추정 계수 값을 바탕으로 Test Set에서 예측한 후 mean squared error 값을 보고한다.

  Mean squared error : 38.3268180861662

부록

PART1

```
# Part1

# Q1

# Q2
from pandas.io.stata import precision_loss_doc
import requests
from bs4 import BeautifulSoup
import pandas as pd

url = 'https://www.skysports.com/premier-league-table/2019'
page = requests.get(url)
soup = BeautifulSoup(page.text, 'html.parser')
table = soup.find('table', {"class" : 'standing-table__table callfn'})

headers= []
for i in table.find_all('th') :
    title = i.text.strip()
    headers.append(title)

df = pd.DataFrame(columns = headers)


for j in table.find_all('tr')[1:] :
    row_data = j.find_all('td')
    row = [tr.text for tr in row_data]
    new_row = []
    for a in row :
        a = a.strip('\n')
        new_row.append(a)
    length = len(df)
    df.loc[length] = new_row
```

```
# Q3
df = df.astype({'GD':'int', 'Pl':'int', 'W':'int'})
pl = df.Pl.tolist()
w = df.W.tolist()

win_prob = []
for i in range(len(pl)) :
    x = w[i] / pl[i]
    win_prob.append(x)
df['win_prob'] = win_prob

from sklearn.linear_model import LinearRegression
import numpy as np
from  sklearn.metrics import mean_squared_error

lm = LinearRegression()
X = df[["GD"]]
Y = df["win_prob"]
lm.fit(X, Y)
Yhat = lm.predict(X)

a = (lm.intercept_) # y 절편
b = (lm.coef_) # 기울기
r_squared = (lm.score(X,Y)) # 결정계수

# Q4
import statsmodels.api as sm
X2 = sm.add_constant(X)
est = sm.OLS(Y, X2)
est2 = est.fit()
print(est2.summary())
```

PART2

```
# Part2

from numpy.core.fromnumeric import mean
from sklearn.linear_model import LinearRegression
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
missing_values = ["na", "NaN"]
df = pd.read_csv('boston_csv.csv', na_values = missing_values)
df.dropna(inplace = True)

# Q6
from scipy import stats
```

```
df.describe() # 변수별 요약통계량
df.corr() # 상관관계 파악
ax = sns.heatmap(df) # heatmap 구현
plt.title("Heatmap")
plt.show()

# Q7
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

lm = LinearRegression()
X = df[["LSTAT"]]
Y = df["MEDV"]
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.25)

lm.fit(X_train, Y_train)
Yhat = lm.predict(X_train)

lm.intercept_ # y 절편
lm.coef_ # 기울기
lm.score(X_train, Y_train) # 결정계수
mean_squared_error(Y_train, Yhat) # mean squared error

Yhat_test = lm.predict(X_test)
mean_squared_error(Y_test, Yhat_test)

# Q8
lm = LinearRegression()
Z = df[["LSTAT", "TAX"]]
Y = df["MEDV"]
Z_train, Z_test, Y_train, Y_test = train_test_split(Z, Y, test_size=0.25)

lm.fit(Z_train, Y_train)
Yhat = lm.predict(Z_train)

lm.intercept_ # y 절편
lm.coef_ # 기울기
lm.score(Z_train, Y_train) # 결정계수
mean_squared_error(Y_train, Yhat) # mean squared error

Yhat_test = lm.predict(Z_test)
mean_squared_error(Y_test, Yhat_test)
```