다음 대선과 총선 예측

김수민

절차

- 데이터 수집, 목표값
- 빈 데이터 채우기
- 딥러닝 후 예측값

데이터 수집과정

- 출처 - 위키백과, 나무위키
- 대통령 분류 기준
  - 0 = 여당
  - 1 = 야당
- 국회의원 분류 기준
  - 민주당계
  - 보수당계
  - 진보당계
  - 무소속

국회의원 분류법

- 해당 모델에만 유효
- 선거 당시의 의석수 기준
- 추후 흡수된 주요 세력의 정착당

국회의원 분류 예

신민당(67년) 창당

민주공화당(박정희)과 대립 -> 민주당계

80년 강제해산

85년 재창당

87년 통일민주당, 김대중 탈당

90년 3당 합당 - > 보수당계

연도	국세/단위 억원	성장률	물가 상승률	언론자유도	대통령 투표율	국회의원 투표율	대통령 여/야	민주당계	보수당계	진보당계	무소속
1966	951	12	11.3								
1967		9.1	10.9		83.6	76.1	0	46	129	0	0
1968		13.2	10.8								
1969		14.6	12.4								
1970	3,648	10.1	16								
1971		10.5	13.5		79.8	73.2	0	90	114	0	0
1972		7.2	11.7								
1973		14.9	3.2			73		54	146	0	19
1974		9.5	24.3								
1975		7.8	25.2								
1976		13.2	15.3								
1977		12.3	10.1								
1978		11	14.5			77.1		64	145	0	22
1979		8.7	18.3								
1980	58,077	-1.6	28.7								
1981		7.2	21.4			78.4		84	177	4	11
1982		8.3	7.2								
1983		13.4	3.4								
1984		10.6	2.3								
1985	118,764	7.8	2.5			84.6		103	168	1	4
1986	136,063	11.3	2.8								
1987	163,437	12.7	3		89.2		0				
1988	194,842	12	7.1			75.8		130	160	0	9
1989	212,341	7.1	5.7								

2004	1,177,957	5.2	3.6	29		60.6		162	135	0	
2005	1,274,657	4.3	2.8	30							
2006	1,380,443	5.3	2.2	30							
2007	1,614,591	5.8	2.5	30	63		1				
2008	1,673,060	3	4.7	3		46.1		81	188	5	
2009	1,645,407	0.8	2.8	30							
2010	1,777,184	6.8	2.9	32							
2011	1,923,812	3.7	4	32							
2012	2,030,149	2.4	2.2	31	75.8	54.2	0	127	157	13	
2013	2,019,065	3.2	1.3	32							
2014	2,055,198	3.2	1.3	33							
2015	2,178,851	2.8	0.7	33							
2016	2,425,617	2.9	1	34		58		123	160	6	
2017	2,653,849	3.2	1.9		77.2		1				
2018	2,935,704	2.9	1.5								
2019	2,934,543	2.2	0.4								
2020	2,855,462	-0.7	0.5			66.2		183	106	6	
2021	2989668.71	4.1	2.5								
2022	3130183.14	2.5	4.8		77.08		1				
2023	3277301.75	2.9	3								
2024	3431334.93										
2025	3592607.68										
2026											
2027		2.3	1.9								

- 포용적 경제회복과 선도국가 도약을 위한 재정의 역할 지속
- 총량관리 강화 및 선제적 위험대응을 통한 재정의 지속가능성 제고
- 과감한 재정혁신을 통한 재정의 효과성·민주성 제고

2) 중기 재정전망 및 재정운용 목표

- (재정수입) '21~'25년 기간 중 연평균 4.7% 증가할 전망
- (재정지출) '21~'25년 기간 중 연평균 5.5% 증가할 전망
- (재정수지) '21년 △4%대(2차 추경)에서 '22~'25년 △2% 중반~△3% 수준으로
- (국가채무) '25년 국가채무비율은 50% 후반 수준 전망

3. 분야별 재원배분 방향

- (사회분야) 코로나19 위기 및 경제사회 구조 전환에 따른 新양극화 해소를 위해 소득
- (경제분야) 한국판 뉴딜 2.0 추진, 미래 전략산업 육성, 국가균형발전 투자 확대 등 지
- (행정분야) 스마트 강군 육성, 국가방역 기반체계 강화 등 국민안전 및 삶의 질 개선을

4. 재정혁신 추진방향

경제e정표 경제정책시계열서비스 (kdi.re.kr) **'21~'25년 국가재정운용계획**

선진국과 신흥국 경제 성장률 예상 (2027년)

국가별 성장률 (IMF 예상)	평균 성장률 (%)		실질 GDP 성장 예상 (%)		
연도	2020 년	2021년	2022년	2023년	2027 년
전 세계 성장률	3.3	6.1	4.4	3.6	3.3
선진국 경제 전체 평균	4.7	5.2	3.6	2.4	1.6
미국	3.5	5.7	3.5	2.3	1.7
유로 지역	6.6	5.3	3.8	2.3	1.3
일본	4.8	1.6	2.5	2.3	0.4
신흥국 시장 및 경제 전체 평균	2.2	6.8	5	4.4	4.3
신흥국 및 개발도상국(아시아 전체) 평균	1	7.3	6	5.6	5.2
중국	2.2	8.1	4.4	5.1	4.8
한국	0.9	4	2.5	2.9	2.3
인도	6.6	8.9	8.2	6.9	6.2
인도네시아	2.1	3.7	5.4	6	5.2
베트남	2.9	2.6	6	7.2	6.7
필리핀	9.6	5.6	6.5	6.3	6.5
이집트	3.6	3.3	5.9	5	5.9

세계경제 전망, 아시아 신흥국 연평균 5~6% 경제 성장률 전망(IMF) (tistory.com)

Table A5. Summary of Inflation

(Percent)

	Average 2004–13	2014	2015	2016	2017	2018	2019	2020	2021	Projections 2022	2023	2027
GDP Deflators												
Advanced Economies	**1.6**	**1.4**	**1.3**	**1.0**	**1.5**	**1.7**	**1.5**	**1.4**	**2.9**	**4.4**	**2.4**	**1.8**
United States	2.1	1.9	1.0	1.0	1.9	2.4	1.8	1.2	4.2	6.3	3.0	2.0
Euro Area	1.6	0.9	1.4	0.9	1.1	1.5	1.7	1.6	2.0	3.2	2.4	1.9
Japan	–1.0	1.7	2.1	0.4	–0.1	0.0	0.6	0.9	–0.9	0.4	0.4	0.5
Other Advanced Economies[1]	2.0	1.3	1.1	1.2	1.9	1.7	1.2	1.8	3.6	4.4	2.3	2.0
Consumer Prices												
Advanced Economies	**2.0**	**1.4**	**0.3**	**0.7**	**1.7**	**2.0**	**1.4**	**0.7**	**3.1**	**5.7**	**2.5**	**1.9**
United States	2.4	1.6	0.1	1.3	2.1	2.4	1.8	1.2	4.7	7.7	2.9	2.0
Euro Area[2]	2.0	0.4	0.2	0.2	1.5	1.8	1.2	0.3	2.6	5.3	2.3	1.9
Japan	–0.1	2.8	0.8	–0.1	0.5	1.0	0.5	0.0	–0.3	1.0	0.8	1.0
Other Advanced Economies[1]	2.3	1.5	0.5	0.9	1.8	1.9	1.4	0.6	2.5	4.8	3.0	1.9
Emerging Market and Developing Economies[3]	**6.3**	**4.7**	**4.7**	**4.3**	**4.4**	**4.9**	**5.1**	**5.2**	**5.9**	**8.7**	**6.5**	**4.1**
Regional Groups												
Emerging and Developing Asia	5.0	3.4	2.7	2.8	2.4	2.7	3.3	3.1	2.2	3.5	2.9	2.7
Emerging and Developing Europe	8.1	6.5	10.6	5.5	5.6	6.4	6.6	5.3	9.5	27.1	18.1	6.8
Latin America and the Caribbean	4.9	4.9	5.4	5.5	6.3	6.6	7.7	6.4	9.8	11.2	8.0	5.0
Middle East and Central Asia	8.4	6.5	5.6	5.7	6.9	9.8	7.8	10.6	13.2	12.8	10.5	6.9
Sub-Saharan Africa	8.6	6.4	6.7	10.3	10.6	8.3	8.1	10.2	11.0	12.2	9.6	6.7

세계경제 전망, 아시아 신흥국 연평균 5~6% 경제 성장률 전망(IMF) (tistory.com)

2004	1,177,957	5.2	3.6	29		60.6		162	135	0
2005	1,274,657	4.3	2.8	30						
2006	1,380,443	5.3	2.2	30						
2007	1,614,591	5.8	2.5	30	63		1			
2008	1,673,060	3	4.7	3		46.1		81	188	5
2009	1,645,407	0.8	2.8	30						
2010	1,777,184	6.8	2.9	32						
2011	1,923,812	3.7	4	32						
2012	2,030,149	2.4	2.2	31	75.8	54.2	0	127	157	13
2013	2,019,065	3.2	1.3	32						
2014	2,055,198	3.2	1.3	33						
2015	2,178,851	2.8	0.7	33						
2016	2,425,617	2.9	1	34		58		123	160	6
2017	2,653,849	3.2	1.9		77.2		1			
2018	2,935,704	2.9	1.5							
2019	2,934,543	2.2	0.4							
2020	2,855,462	-0.7	0.5			66.2		183	106	6
2021	2989668.71	4.1	2.5							
2022	3130183.14	2.5	4.8		77.08		1			
2023	3277301.75	2.9	3							
2024	3431334.93									
2025	3592607.68									
2026										
2027		2.3	1.9							

빈 데이터 채우기

- 평균
- 고정값 – 대통령 여/야
- 회귀예측 - 언론자유도

```
print(data.iloc[56:62,5:6])
#대통령 투표율 완료 국회의원 투표율 및 분류별 국회의원 수
data.iloc[0:1,6:7] = data.iloc[1:2,6:7]
data.iloc[0:1,8:] = data.iloc[1:2,8:]
print(data.loc[:,['국회의원 투표율','민주당계','보수당계','진보당계','무소속']])

data.loc[[2,3,4],['국회의원 투표율','민주당계','보수당계','진보당계','무소속']] = \
data.loc[[2,3,4],['국회의원 투표율','민주당계','보수당계','진보당계','무소속']].fillna(\
data.loc[[1,5],['국회의원 투표율','민주당계','보수당계','진보당계','무소속']].mean())

data.loc[[6],['국회의원 투표율','민주당계','보수당계','진보당계','무소속']] = data.loc[[
['국회의원 투표율','민주당계','보수당계','진보당계','무소속']].mean())

data.loc[[8,9,10,11],['국회의원 투표율','민주당계','보수당계','진보당계','무소속']] = da
(data.loc[[7,12],['국회의원 투표율','민주당계','보수당계','진보당계','무소속']].mean())

data.loc[[13,14],['국회의원 투표율','민주당계','보수당계','진보당계','무소속']] = data.l
[[12,15],['국회의원 투표율','민주당계','보수당계','진보당계','무소속']].mean())

data.loc[[16,17,18],['국회의원 투표율','민주당계','보수당계','진보당계','무소속']] = dat
```

연도	국세/단위 억원	성장률	물가 상승률	언론자유도	대통령 투표율	국회의원 투표율	대통령 여/야	민주당계	보수당계	진보당계	무소속
1966	951	12	11.3								
1967		9.1	10.9		83.6	76.1	0	46	129	0	0
1968		13.2	10.8								
1969		14.6	12.4								
1970	3,648	10.1	16								
1971		10.5	13.5		79.8	73.2	0	90	114	0	0
1972		7.2	11.7								
1973		14.9	3.2			73		54	146	0	19
1974		9.5	24.3								
1975		7.8	25.2								
1976		13.2	15.3								
1977		12.3	10.1								
1978		11	14.5			77.1		64	145	0	22
1979		8.7	18.3								
1980	58,077	-1.6	28.7								
1981		7.2	21.4			78.4		84	177	4	11
1982		8.3	7.2								
1983		13.4	3.4								
1984		10.6	2.3								
1985	118,764	7.8	2.5			84.6		103	168	1	4
1986	136,063	11.3	2.8								
1987	163,437	12.7	3		89.2		0				
1988	194,842	12	7.1			75.8		130	160	0	9
1989	212,341	7.1	5.7								

당선시 기준

이전 4년 동일값

```
#대통령 여/야
data.iloc[0:26,7:8] = 0
data.iloc[32:36,7:8] = 0
data.iloc[42:46,7:8] = 0

data.iloc[27:31,7:8] = 1
data.iloc[37:41,7:8] = 1
data.iloc[47:56,7:8] = 1
print(data)
```

언론 자유도

- 회귀예측
  - 경제성장률
  - 물가 상승률

```
from sklearn import linear_model
reg_lin = linear_model.LinearRegression()
x = data.iloc[27:51,2:4]
y = data['언론자유도']
y = y.dropna()
print(x.shape, y.shape)
reg_lin.fit(x,y)
# pressfree = reg_lin.predict(data.iloc[:27,2:4])
# print(pressfree.shape)
print(data['언론자유도'])
data.iloc[0:1,4:5] = reg_lin.predict(data.iloc[0:1,2:4])
data.iloc[1:2,4:5] = reg_lin.predict(data.iloc[1:2,2:4])
data.iloc[2:3,4:5] = reg_lin.predict(data.iloc[2:3,2:4])
data.iloc[3:4,4:5] = reg_lin.predict(data.iloc[3:4,2:4])
```

2004	1,177,957	5.2	3.6	29		60.6		162	135	0	
2005	1,274,657	4.3	2.8	30							
2006	1,380,443	5.3	2.2	30							
2007	1,614,591	5.8	2.5	30	63		1				
2008	1,673,060	3	4.7	3		46.1		81	188	5	
2009	1,645,407	0.8	2.8	30							
2010	1,777,184	6.8	2.9	32							
2011	1,923,812	3.7	4	32							
2012	2,030,149	2.4	2.2	31	75.8	54.2	0	127	157	13	
2013	2,019,065	3.2	1.3	32							
2014	2,055,198	3.2	1.3	33							
2015	2,178,851	2.8	0.7	33							
2016	2,425,617	2.9	1	34		58		123	160	6	
2017	2,653,849	3.2	1.9		77.2		1				
2018	2,935,704	2.9	1.5								
2019	2,934,543	2.2	0.4								
2020	2,855,462	-0.7	0.5			66.2		183	106	6	
2021	2989668.71	4.1	2.5								
2022	3130183.14	2.5	4.8		77.08		1				
2023	3277301.75	2.9	3								
2024	3431334.93										
2025	3592607.68										
2026											
2027		2.3	1.9								

하한값

```
Name: 언론자유도, dtype: float64
      연도     국세/단위 억원   성장률  물가 상승률
    보수당계  진보당계   무소속
0   1966      951.000  12.0   11.30  14.939650  ...
1   1967     2299.500   9.1   10.90  16.308046  ...
2   1968     2299.500  13.2   10.80  15.325226  ...
3   1969     2299.500  14.6   12.40  12.652506  ...
4   1970     3648.000  10.1   16.00   8.739532  ...
5   1971    30862.500  10.5   13.50  12.205145  ...
6   1972    30862.500   7.2   11.70  15.685631  ...
7   1973    30862.500  14.9    3.20  25.727785  ...
8   1974    30862.500   9.5   24.30  -2.966207  ...
9   1975    30862.500   7.8   25.20  -3.786559  ...
10  1976    30862.500  13.2   15.30   8.389415  ...
11  1977    30862.500  12.3   10.10  16.573487  ...
12  1978    30862.500  11.0   14.50  10.637667  ...
13  1979    30862.500   8.7   18.30   5.834550  ...
14  1980    58077.000  -1.6   28.70  -6.211001  ...
```

```
data.loc[data['언론자유도']<10,'언론자유도'] = 10
print(data)
```

1966	951.000	12.0	11.30	14.939650	83.60	76.10	0.0	46.0	129.0	0.0	0.0
1967	2299.500	9.1	10.90	16.308046	83.60	76.10	0.0	46.0	129.0	0.0	0.0
1968	2299.500	13.2	10.80	15.325226	81.70	74.65	0.0	68.0	121.5	0.0	0.0
1969	2299.500	14.6	12.40	12.652506	81.70	74.65	0.0	68.0	121.5	0.0	0.0
1970	3648.000	10.1	16.00	10.000000	81.70	74.65	0.0	68.0	121.5	0.0	0.0
1971	30862.500	10.5	13.50	12.205145	79.80	73.20	0.0	90.0	114.0	0.0	0.0
1972	30862.500	7.2	11.70	15.685631	84.50	73.10	0.0	72.0	130.0	0.0	9.5
1973	30862.500	14.9	3.20	25.727785	84.50	73.00	0.0	54.0	146.0	0.0	19.0
1974	30862.500	9.5	24.30	10.000000	84.50	75.05	0.0	59.0	145.5	0.0	20.5
1975	30862.500	7.8	25.20	10.000000	84.50	75.05	0.0	59.0	145.5	0.0	20.5
1976	30862.500	13.2	15.30	10.000000	84.50	75.05	0.0	59.0	145.5	0.0	20.5
1977	30862.500	12.3	10.10	16.573487	84.50	75.05	0.0	59.0	145.5	0.0	20.5
1978	30862.500	11.0	14.50	10.637667	84.50	77.10	0.0	64.0	145.0	0.0	22.0
1979	30862.500	8.7	18.30	10.000000	84.50	77.75	0.0	74.0	161.0	2.0	16.5
1980	58077.000	-1.6	28.70	10.000000	84.50	77.75	0.0	74.0	161.0	2.0	16.5
1981	88420.500	7.2	21.40	10.000000	84.50	78.40	0.0	84.0	177.0	4.0	11.0
1982	88420.500	8.3	7.20	21.819388	84.50	81.50	0.0	93.5	172.5	2.5	7.5
1983	88420.500	13.4	3.40	25.853641	84.50	81.50	0.0	93.5	172.5	2.5	7.5
1984	88420.500	10.6	2.30	28.195704	84.50	81.50	0.0	93.5	172.5	2.5	7.5
1985	118764.000	7.8	2.50	28.678532	84.50	84.60	0.0	103.0	168.0	1.0	4.0
1986	136063.000	11.3	2.80	27.288398	84.50	80.20	0.0	116.5	164.0	0.5	6.5
1987	163437.000	12.7	3.00	26.617929	89.20	80.20	0.0	116.5	164.0	0.5	6.5
1988	194842.000	12.0	7.10	20.946406	85.55	75.80	0.0	130.0	160.0	0.0	9.0
1989	212341.000	7.1	5.70	24.294172	85.55	73.85	0.0	114.0	170.0	0.0	15.0
1990	268474.000	9.9	8.60	19.377785	85.55	73.85	0.0	114.0	170.0	0.0	15.0

1991	303198.000	10.8	9.30	18.129523	85.55	73.85	0.0	114.0	170.0	0.0	15.0
1992	352184.000	6.2	6.20	23.826217	81.90	71.90	0.0	98.0	180.0	0.0	21.0
1993	392606.000	6.9	4.80	29.000000	81.30	67.90	1.0	112.5	168.5	0.0	18.0
1994	472617.000	9.3	6.30	28.000000	81.30	67.90	1.0	112.5	168.5	0.0	18.0
1995	567745.000	9.6	4.50	22.000000	81.30	67.90	1.0	112.5	168.5	0.0	18.0
1996	649602.000	7.9	4.90	25.000000	81.30	63.90	1.0	127.0	157.0	0.0	15.0
1997	699277.000	6.2	4.40	28.000000	80.70	60.55	1.0	121.0	155.0	0.0	10.0
1998	677977.000	-5.1	7.50	28.000000	75.75	60.55	0.0	121.0	155.0	0.0	10.0
1999	756580.000	11.5	0.80	27.000000	75.75	60.55	0.0	121.0	155.0	0.0	10.0
2000	929347.000	9.1	2.30	27.000000	75.75	57.20	0.0	115.0	153.0	0.0	5.0
2001	957928.000	4.9	4.10	30.000000	75.75	58.90	0.0	138.5	144.0	0.0	3.5
2002	1039678.000	7.7	2.80	29.000000	70.80	58.90	0.0	138.5	144.0	0.0	3.5
2003	1146642.000	3.1	3.50	29.000000	66.90	58.90	1.0	138.5	144.0	0.0	3.5
2004	1177957.000	5.2	3.60	29.000000	66.90	60.60	1.0	162.0	135.0	0.0	2.0
2005	1274657.000	4.3	2.80	30.000000	66.90	53.35	1.0	121.5	161.5	2.5	13.5
2006	1380443.000	5.3	2.20	30.000000	66.90	53.35	1.0	121.5	161.5	2.5	13.5
2007	1614591.000	5.8	2.50	30.000000	63.00	53.35	1.0	121.5	161.5	2.5	13.5
2008	1673060.000	3.0	4.70	10.000000	69.40	46.10	0.0	81.0	188.0	5.0	25.0
2009	1645407.000	0.8	2.80	30.000000	69.40	50.15	0.0	104.0	172.5	9.0	14.0
2010	1777184.000	6.8	2.90	32.000000	69.40	50.15	0.0	104.0	172.5	9.0	14.0
2011	1923812.000	3.7	4.00	32.000000	69.40	50.15	0.0	104.0	172.5	9.0	14.0
2012	2030149.000	2.4	2.20	31.000000	75.80	54.20	0.0	127.0	157.0	13.0	3.0
2013	2019065.000	3.2	1.30	32.000000	76.50	56.10	1.0	125.0	158.5	9.5	7.0
2014	2055198.000	3.2	1.30	33.000000	76.50	56.10	1.0	125.0	158.5	9.5	7.0
2015	2178851.000	2.8	0.70	33.000000	76.50	56.10	1.0	125.0	158.5	9.5	7.0

2003	1146642.000	3.1	3.50	29.000000	66.90	58.90	1.0	138.5	144.0	0.0	3.5
2004	1177957.000	5.2	3.60	29.000000	66.90	60.60	1.0	162.0	135.0	0.0	2.0
2005	1274657.000	4.3	2.80	30.000000	66.90	53.35	1.0	121.5	161.5	2.5	13.5
2006	1380443.000	5.3	2.20	30.000000	66.90	53.35	1.0	121.5	161.5	2.5	13.5
2007	1614591.000	5.8	2.50	30.000000	63.00	53.35	1.0	121.5	161.5	2.5	13.5
2008	1673060.000	3.0	4.70	10.000000	69.40	46.10	0.0	81.0	188.0	5.0	25.0
2009	1645407.000	0.8	2.80	30.000000	69.40	50.15	0.0	104.0	172.5	9.0	14.0
2010	1777184.000	6.8	2.90	32.000000	69.40	50.15	0.0	104.0	172.5	9.0	14.0
2011	1923812.000	3.7	4.00	32.000000	69.40	50.15	0.0	104.0	172.5	9.0	14.0
2012	2030149.000	2.4	2.20	31.000000	75.80	54.20	0.0	127.0	157.0	13.0	3.0
2013	2019065.000	3.2	1.30	32.000000	76.50	56.10	1.0	125.0	158.5	9.5	7.0
2014	2055198.000	3.2	1.30	33.000000	76.50	56.10	1.0	125.0	158.5	9.5	7.0
2015	2178851.000	2.8	0.70	33.000000	76.50	56.10	1.0	125.0	158.5	9.5	7.0
2016	2425617.000	2.9	1.00	34.000000	76.50	58.00	1.0	123.0	160.0	6.0	11.0
2017	2653849.000	3.2	1.90	30.799775	77.20	62.10	1.0	153.0	133.0	6.0	8.0
2018	2935704.000	2.9	1.50	31.454225	77.14	62.10	1.0	153.0	133.0	6.0	8.0
2019	2934543.000	2.2	0.40	33.219640	77.14	62.10	1.0	153.0	133.0	6.0	8.0
2020	2855462.000	-0.7	0.50	33.872946	77.14	66.20	1.0	183.0	106.0	6.0	5.0
2021	2989668.714	4.1	2.50	29.694532	77.14	66.20	1.0	183.0	106.0	6.0	5.0
2022	3130183.144	2.5	4.80	26.844468	77.08	66.20	1.0	183.0	106.0	6.0	5.0
2023	3277301.751	2.9	3.00	29.308955	77.08	66.20	NaN	NaN	NaN	NaN	NaN
2024	3431334.934	2.6	2.45	30.177933	77.08	66.20	NaN	NaN	NaN	NaN	NaN
2025	3592607.675	2.6	2.45	30.177933	77.08	66.20	NaN	NaN	NaN	NaN	NaN
2026	3592607.675	2.6	2.45	30.177933	77.08	66.20	NaN	NaN	NaN	NaN	NaN
2027	3592607.675	2.3	1.90	31.046910	77.08	66.20	NaN	NaN	NaN	NaN	NaN

```python
data = np.load('c:/project/개인1/npy, weight 저장/2/data.npy')
# print(data)
print(data.shape)
print(len(data))

def split_x(dataset, size):
    aaa = []
    for i in range(len(dataset)-4):
        subset = dataset[i:i+size,1:7]
        aaa.append(subset)
    print(type(aaa))
    return np.array(aaa)


x = split_x(data,5)
```

```
import torch
from torch import nn
import torch.nn.functional as F
econo_x_train = torch.from_numpy(econo_x_train)
econo_x_test = torch.from_numpy(econo_x_test)
econo_x_pred = torch.from_numpy(econo_x_pred)
democ_x_train = torch.from_numpy(democ_x_train)
democ_x_test = torch.from_numpy(democ_x_test)
democ_x_pred = torch.from_numpy(democ_x_pred)
econo_x_train = F.interpolate(econo_x_train, scale_factor=50, mode='linear')
econo_x_test = F.interpolate(econo_x_test, scale_factor=50, mode='linear')
econo_x_pred = F.interpolate(econo_x_pred, scale_factor=50, mode='linear')
democ_x_train = F.interpolate(democ_x_train, scale_factor=50, mode='linear')
democ_x_test = F.interpolate(democ_x_test, scale_factor=50, mode='linear')
democ_x_pred = F.interpolate(democ_x_pred, scale_factor=50, mode='linear')
econo_x_train = econo_x_train.numpy()
econo_x_test = econo_x_test.numpy()
econo_x_pred = econo_x_pred.numpy()
democ_x_train = democ_x_train.numpy()
```

```
#앙상블모델 구성
from keras.models import Model, load_model
from keras.layers import Input, Dense, Conv1D, Conv2D, Fl
GRU, concatenate,Dropout,MaxPooling1D,MaxPooling2D
from keras.callbacks import EarlyStopping, ModelCheckpoin
es = EarlyStopping(monitor='val_loss', patience=50000,mod
restore_best_weights=True,verbose=1)
mc = ModelCheckpoint('bestmodel.h5',monitor='val_loss',mo
#모델1
econo = Input(shape=(5,3))
econo = Dense(200,activation='relu')(econo)
econo = Dense(15)(econo)
econo = Reshape((75,1))(econo)
econo = ReLU(2800,5)(econo)
econo = LSTM(64,activation='relu')(econo)
# econo = Flatten()(econo)
econo = Dense(200)(econo)
econo = Dense(200)(econo)
econo = Dense(200)(econo)
econo = Dense(200)(econo)
econo = Dense(15)(econo)
econo = Reshape((5,3))(econo)
# econo  = Flatten()(econo)
# econo  = Dense(200)(econo)
# econo = Reshape((50,4))(econo)
# econo =  ReLU(8,10)(econo)

#모델2
```

```
econo = Dense(200)(econo)
econo = Dense(15)(econo)
econo = Reshape((5,3))(econo)
# econo  = Flatten()(econo)
# econo  = Dense(200)(econo)
# econo = Reshape((50,4))(econo)
# econo =  ReLU(8,10)(econo)

#모델2
democ = Input(shape=(5,3))
democ = Dense(200)(democ)
democ = Dense(200)(democ)
democ = Dense(200)(democ)
democ = Dense(15)(democ)
democ = Reshape((75,1))(democ)
democ = ReLU(28,10)(democ)
democ = LSTM(64,return_sequences=True)(democ)
democ = Flatten()(democ)
democ = Dense(2000, activation='relu')(democ)
democ = Dense(15)(democ)
democ = Reshape((5,3))(democ)

# democ  = Flatten()(democ)
# democ  = Dense(200)(democ)
# democ = Reshape((50,4))(democ)
# democ =  ReLU(8,10)(democ)
```

```
#merge
president = concatenate((econo,democ))
president = Conv1D(12,3)(president)
president = Flatten()(president)
president = Dense(200,activation='tanh')(president)
president = Dense(200,activation='tanh')(president)
president = Dropout(0.2)(president)
president = Dense(200,activation='elu')(president)
president = Dense(20)(president)
president = Dense(1,activation='sigmoid')(president)

congress = concatenate((econo,democ))
congress = Conv1D(12,3)(congress)
congress = Flatten()(congress)
congress = Dense(800,activation='relu')(congress)
congress = Dense(800,activation='relu')(congress)
congress = Dropout(0.4)(congress)
congress = Dense(80)(congress)
congress = Dense(4)(congress)
```

```
congress = Reshape((5,5))(congress)
congress = LSTM(4,return_sequences=True,activation='relu')(congress)

model = Model(inputs=[econo,democ], outputs=[president,congress],)
model.summary()

# model.load_weights('c:/project/개인1/npy, weight 저장/2/weight.h5')

model.compile(loss=['binary_crossentropy','mae'], optimizer='AdaMax')
hist = model.fit([econo_x_train,democ_x_train],[president_y_train,congressmember_y_train],
epochs=30000,batch_size=10,
                 validation_split=0.1,
                 callbacks=[es,mc])

model.save_weights('c:/project/개인1/npy, weight 저장/2/weight.h5')


loss = model.evaluate([econo_x_test,democ_x_test],[president_y_test,congressmember_y_test])
# pred_presi, pred_congress = model.predict([econo_x_test, democ_x_test])
```

```
# print(f'r2',congress_r2_score)
# # print(pred.shape)
# # print(presi_r2_score)
# # result = round(pred)
# # print(np.round(pred,))
pred_presi, pred_congress = model.predict([econo_x_pred,democ_x_pred])


print(np.where(pred_presi[0][-1]>=0.5,'27년 대선에서는 야당후보가 당선됩니다','27년 대선에서는
여당후보가 당선됩니다'))
# print(pred[1][-4].round())
total_congress = pred_congress[1][-4][0].round()+pred_congress[1][-4][1].round()+pred_congress[1]
[-4][2].round()+pred_congress[1][-4][3].round()
print('24년 총선에서 민주당계는',(pred_congress[1][-4][0].round()).astype(int),'명,','보수당계는',
(pred_congress[1][-4][1].round()).astype(int),'명,','진보당계',(pred_congress[1][-4][2].round()).
astype(int),'명,','무소속', (pred_congress[1][-4][3].round()).astype(int),'명이 당선 됩니다.')
print('합계', (total_congress).astype(int),'명')

# print(pred.shape)

print()
```

```
print(president_y)
print(congressmember_y)
print(president_y.shape, congressmember_y.shape)

from sklearn.model_selection import train_test_split
econo_x_train, econo_x_test, democ_x_train, democ_x_test, president_y_train, president_
congressmember_y_train, congressmember_y_test = train_test_split(econo_x, democ_x,presi
```

문제 출력 디버그 콘솔 터미널 JUPYTER 2: Python Debu

```
 [1.2266479e+02 1.5914619e+02 0.0000000e+00 1.0632865e+01]
 [1.2266479e+02 1.5914619e+02 0.0000000e+00 1.0632865e+01]
 [1.2266479e+02 1.5914619e+02 0.0000000e+00 1.0632865e+01]
 [1.2266479e+02 1.5914619e+02 0.0000000e+00 1.0632865e+01]]

[[4.5725342e-28 1.5914619e+02 0.0000000e+00 0.0000000e+00]
 [1.2266479e+02 1.5914619e+02 0.0000000e+00 1.0632865e+01]
 [1.2266479e+02 1.5914619e+02 0.0000000e+00 1.0632865e+01]
 [1.2266479e+02 1.5914619e+02 0.0000000e+00 1.0632865e+01]
 [1.2266479e+02 1.5914619e+02 0.0000000e+00 1.0632865e+01]]]
27년 대선에서는 야당후보가 당선됩니다
24년 총선에서 민주당계는 123 명, 보수당계는 159 명, 진보당계 0 명, 무소속 11 명이 당선 됩니다.
합계 293 명

PS C:\others\3uxeca>
```

핵심코드

- `import torch.nn.functional as F`
- `econo_x_test = F.interpolate(econo_x_test, scale_factor=50, mode='linear')`

- 적은 표본 수 -> 충분한 양으로 오버샘플링

질문 사항

```
Roscaler = RobustScaler()
Norscaler = Normalizer()

econo_x_train = econo_x_train.reshape(-1,15)
econo_x_test = econo_x_test.reshape(-1,15)
econo_x_pred = econo_x_pred.reshape(-1,15)

democ_x_train = democ_x_train.reshape(-1,15)
democ_x_test = democ_x_test.reshape(-1,15)
democ_x_pred = democ_x_pred.reshape(-1,15)

econo_x_train[:,0:1] = Roscaler.fit_transform(econo_x_train[:,0:1])
econo_x_test[:,0:1] = Roscaler.transform(econo_x_test[:,0:1])
econo_x_pred[:,0:1] = Roscaler.transform(econo_x_pred[:,0:1])

econo_x_train[:,1:3] = Stanscaler.fit_transform(econo_x_train[:,1:3])
econo_x_test[:,1:3] = Stanscaler.transform(econo_x_test[:,1:3])
econo_x_pred[:,1:3] = Stanscaler.transform(econo_x_pred[:,1:3])
```

핵심코드 2

- `data[:,1:2] = Roscaler.fit_transform(data[:,1:2])`
- `data[:,2:4] = Stanscaler.fit_transform(data[:,2:4])`
- `data[:,4:5] = Maxscaler.fit_transform(data[:,4:5])`
- `data[:,6:8] = Minscaler.fit_transform(data[:,6:8])`

  - 비슷한 분표를 보이는 데이터를 묶어서 처리

평균 채움

- 경제 모델 - 국세 수입, 성장률, 물가 상승률
- 민주화 모델 - 대통령, 국회의원 투표율
- Y값 - 분류별 국회의원 수

R2 스코어