

# 빅데이터 처리 기말 프로젝트

컴퓨터정보과 3-B 202144055 박정원



제출일	2023.12.3	전공	컴퓨터정보과
과목	빅데이터 처리	학번	202144055
담당교수	민정혜	이름	박정원

# 목차

## 1. 주제 선정

1인당 국민 총소득과 연도별 암 발생 지표 간 상관관계 및 앞으로의 암 발생 추이 예상

## 2. 데이터 수집

- 1) 국립 암 센터 암 발생 통계 정보 CSV
- 2) KOICA 오픈 데이터 포털 1인당 GDP CSV

1번 데이터는 데이터 전처리 과정이 필요한 데이터였으며 2번 데이터는 데이터 전처리 과정이 필요하지 않는 데이터였습니다.

## 3. 데이터 가공/정제(데이터 전처리)

1번 데이터인 국립 암 센터 암 발생 통계 정보 csv의 df.head(5)입니다.

	발생연도	성별	국제질병분류	암종	연령군	발생자수	조발생률
0	1999	남녀전체	00. All cancers	모든암	00-04세	538	16.1
1	1999	남녀전체	00. All cancers	모든암	05-09세	297	8.6
2	1999	남녀전체	00. All cancers	모든암	10-14세	306	9.6
3	1999	남녀전체	00. All cancers	모든암	15-19세	527	13.1
4	1999	남녀전체	00. All cancers	모든암	20-24세	719	18.3

해당 데이터를 전처리 하여 머신러닝이 가능한 상태로 맞춰보겠습니다.

## 4. 데이터 시각화

먼저 1번 데이터를 모두 전처리 한 후 연도별 평균 암 조발생률 그래프를 작성하고 애니메이션이 포함된 연도별 암종별 조발생률을 나타내는 그래프 두가지를 시각화 해보고자합니다.

## 5. 데이터 분석(머신러닝)

1번 데이터를 머신러닝하여 현재는 2020년까지의 정보만을 가지고 있지만 앞으로의 5년인 2025년까지의 예상 조발생률을 예측해보고자합니다.

## 6. 분석 결과 및 결론

머신러닝으로 예측한 결과를 포함한 그래프와 2번 데이터의 그래프를 비교하여 서로 상관관계가 있는지를 파악해보고 상관관계가 있다면 있는대로 없으면 왜 상관관계가 없었는지에 대해 결론을 지어보려합니다.

# 주제 선정


## 1. 주제 선정 이유

제가 이번 주제를 선정하게 된 이유는 평소에 의료 분야에 관심을 가지고 있었기도 하였고 많은 사람들이 암 때문에 많은 고통을 받고 있다는 것을 알고 있기에 실제 암 조발생률은 어느정도일까? 와 어떤 암들이 사람들을 괴롭히고 있는 것일까? 에 대한 궁금증을 가지고 해당 주제를 선정해보았습니다.

또한 저는 국민 총 소득이 높아지면 암 조발생률이 떨어질 것이라고 생각하였습니다. 그 이유는 국민 총 소득이 높아지면 주거 환경이 좋아지고 먹는 음식의 질도 좋아질 것이며 의료 산업 역시 성장할 것이기 때문에 국민 총 소득이 높아지면 암 조발생률이 떨어질 것이다. 라는 가정을 가지고 이번 프로젝트를 진행하였습니다.

```
[ ] # 빅데이터 처리 기말 프로젝트
    # 컴퓨터정보과 3-B 202144055 박정원
    # 주제 : 1인당 국민 총소득과 연도별 암 발생 지표 간 비교 및 앞으로의 암 발생 추이 예상

    # 전체 과정 : 데이터 수집 -> 데이터 가공/정제 -> 데이터 시각화 -> 데이터 분석(머신러닝) -> 분석 결과
```

 Cancer.csv

 Real\_Income\_data.csv

1. Cancer.csv : 국립 암 센터 암 발생 통계 정보 CSV
2. Real\_Income\_data.csv : KOICA 오픈 데이터 포털 1인당 GDP CSV

# 데이터 전처리

## 1. 한국어 폰트 설치

```
[ ] # 한국어 폰트 설치
!sudo apt-get install -y fonts-nanum
!sudo fc-cache -fv
!rm ~/.cache/matplotlib -rf

Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
fonts-nanum is already the newest version (20200506-1).
0 upgraded, 0 newly installed, 0 to remove and 15 not upgraded.
/usr/share/fonts: caching, new cache contents: 0 fonts, 1 dirs
/usr/share/fonts/truetype: caching, new cache contents: 0 fonts, 3 dirs
/usr/share/fonts/truetype/humor-sans: caching, new cache contents: 1 fonts, 0 dirs
/usr/share/fonts/truetype/liberation: caching, new cache contents: 16 fonts, 0 dirs
/usr/share/fonts/truetype/nanum: caching, new cache contents: 12 fonts, 0 dirs
/usr/local/share/fonts: caching, new cache contents: 0 fonts, 0 dirs
/root/.local/share/fonts: skipping, no such directory
/root/.fonts: skipping, no such directory
/usr/share/fonts/truetype: skipping, looped directory detected
/usr/share/fonts/truetype/humor-sans: skipping, looped directory detected
/usr/share/fonts/truetype/liberation: skipping, looped directory detected
/usr/share/fonts/truetype/nanum: skipping, looped directory detected
/var/cache/fontconfig: cleaning cache directory
/root/.cache/fontconfig: not cleaning non-existent cache directory
/root/.fontconfig: not cleaning non-existent cache directory
fc-cache: succeeded
```

```
[ ] # 런타임 재시작
! kill -9 $(ps -A | grep python | awk '{print $1}')
```

추후 그래프 시각화 시 폰트가 깨져서 나오기에 사전에 폰트 설치 및 런타임 재시작을 해주었습니다.

```
print(df.isnull().sum())
print('')
print(df.dtypes)
print('')
print(df['성별'].unique())
print('')
print(df['암종'].unique())
print('')
df.info
```

발생연도 0  
성별 0  
국제질병분류 0  
암종 0  
연령군 0  
발생자수 0  
조발생률 5287  
dtype: int64

발생연도 object  
성별 object  
국제질병분류 object  
암종 object  
연령군 object  
발생자수 int64  
조발생률 float64  
dtype: object

['남녀전체' '남자' '여자']

['모든암' '입술, 구강 및 인두' '식도' '위' '대장' '간' '담낭 및 기타담도' '폐' '후두' '폐' '유방'  
'자궁경부' '자궁체부' '난소' '전립선' '고환' '신장' '방광' '뇌 및 중추신경계' '갑상선' '호지킨림프종'  
'비호지킨림프종' '다발성 골수종' '백혈병' '기타 암']

```
<bound method DataFrame.info of
0      1999  남녀전체  00, All cancers  모든암  00-04세  538  16.1
1      1999  남녀전체  00, All cancers  모든암  05-09세  297   8.6
2      1999  남녀전체  00, All cancers  모든암  10-14세  306   9.6
3      1999  남녀전체  00, All cancers  모든암  15-19세  527  13.1
4      1999  남녀전체  00, All cancers  모든암  20-24세  719  18.3

... ..
32770  1999-2020  여자  24, All other cancers  기타 암  70-74세  17114  94.2
32771  1999-2020  여자  24, All other cancers  기타 암  75-79세  18452  132.7
32772  1999-2020  여자  24, All other cancers  기타 암  80-84세  15252  172.2
32773  1999-2020  여자  24, All other cancers  기타 암  85세이상  13278  209.2
32774  1999-2020  여자  24, All other cancers  기타 암  연령전체  143413  26.3

[32775 rows x 7 columns]>
```

먼저 전체 데이터가 정확히 어떻게 구성되어있는지를 확인해보았습니다. 조발생률에서 5287개의 결측치가 존재하고 있었으며 성별에는 남녀전체라는 통합 데이터가 포함되어있었습니다. 암종에는 위와 같이 여러 가지 암들이 있는데 모든암이라는 통합 데이터가 포함되어있습니다. 또한 발생연도에서도 1999-2000의 통합 데이터가 포함되어있었으며 연령군에서도 연령전체라는 통합 데이터가 포함되어있으며 총 32775개의 열과 7개의 행으로 구성되어있음을 알 수 있습니다.

먼저 결측치를 처리해보도록 하겠습니다. 결측치를 처리하는 방법에는 평균치를 대입하거나 예상치를 대입하는 등의 여러 방법들이 있지만 해당 데이터는 사실을 기반으로 한 데이터이기에 평균치를 대입하거나 예상치를 대입하게되면 정보의 무결성이 훼손될 수 있다고 생각하였기 때문에 **결측치를 제거**하는 방향으로 진행하였습니다.

```
[ ] # 결측치 확인: '조발생률' 열에 5287개의 결측치가 있습니다.
    # 그렇다면 결측치를 처리해보겠습니다.

df.dropna(subset=['조발생률'], inplace=True)
print(df.isnull().sum())
```

```
발생연도      0
성별          0
국제질병분류  0
암종          0
연령군        0
발생자수      0
조발생률      0
dtype: int64
```

dropna를 통해서 결측치가 있는 부분을 제거하였습니다. isnull.sum을 통해서 null 값이 있는 데이터프레임의 합을 살펴보니 모두 잘 제거된 모습입니다.

```
[ ] # 데이터 형식 확인 및 조정
df['발생연도'] = df['발생연도'].astype(int)
df['조발생률'] = df['조발생률'].astype(float)

# 필요한 열 선택 및 재구성
df = df[['발생연도', '성별', '암종', '연령군', '발생자수', '조발생률']]

# 데이터 정렬
df = df.sort_values(by=['발생연도', '성별', '연령군'])
```

```
ValueError                                Traceback (most recent call last)
<ipython-input-4-6429593ed21f> in <cell line: 2>()
      1 # 데이터 형식 확인 및 조정
----> 2 df['발생연도'] = df['발생연도'].astype(int)
      3 df['조발생률'] = df['조발생률'].astype(float)
      4
      5 # 필요한 열 선택 및 재구성
```

```
6 frames
/usr/local/lib/python3.10/dist-packages/pandas/core/dtypes/astype.py in astype_nansafe(arr, dtype, copy, skipna)
    169     if copy or is_object_dtype(arr.dtype) or is_object_dtype(dtype):
    169         # Explicit copy, or required since NumPy can't view from / to object.
--> 170         return arr.astype(dtype, copy=True)
    171
    172     return arr.astype(dtype, copy=copy)
```

```
ValueError: invalid literal for int() with base 10: '1999-2020'
```

STACK OVERFLOW 검색

그 후 머신러닝을 위해 데이터 전처리를 진행하던 도중 에러가 발생하였습니다. 그 원인을 알아보니 1999-2020의 데이터 때문임을 알 수 있습니다. 따라서 통합 데이터들을 모두 제거해주어야합니다.

```

# '남녀전체' 데이터를 포함한 행 제거
df = df[df['성별'] != '남녀전체']

# '연령전체' 데이터를 포함한 행 제거
df = df[df['연령군'] != '연령전체']

# '모든암' 데이터를 포함한 행 제거
df = df[df['암종'] != '모든암']

# '발생연도' 열에서 범위 값을 포함하는 행 제거
df = df[~df['발생연도'].str.contains('-')]

# '발생연도' 열을 정수형으로 변환
df['발생연도'] = df['발생연도'].astype(int)

# '조발생률' 열을 실수형으로 변환
df['조발생률'] = df['조발생률'].astype(float)

# 필요한 열 선택 및 재구성
df = df[['발생연도', '성별', '암종', '연령군', '발생자수', '조발생률']]

# 데이터 정렬
df = df.sort_values(by=['발생연도', '성별', '연령군', '암종'])

df.info

```

<bound method DataFrame.info of

	발생연도	성별	암종	연령군	발생자수	조발생률
6574	1999	남자	간	00-04세	11	0.6
19684	1999	남자	고환	00-04세	25	1.4
31483	1999	남자	기타 암	00-04세	79	4.5
23617	1999	남자	뇌 및 중추신경계	00-04세	45	2.6
5263	1999	남자	대장	00-04세	1	0.1
...	...	...	...	...	...	...
16984	2020	여자	자궁체부	85세이상	23	4.0
10429	2020	여자	췌장	85세이상	524	91.9
13051	2020	여자	폐	85세이상	935	163.9
27472	2020	여자	호지킨림프종	85세이상	1	0.2
11740	2020	여자	후두	85세이상	6	1.1

[14902 rows x 6 columns]>

아까 위에서 확인한 '남녀전체', '연령전체', '1999-2000', '모든암' 데이터는 모두 제거하였습니다. 그리고 발생연도를 정수형으로 변환하였고 조발생률은 실수형으로 변환하였습니다. 마지막으로 필요하지 않는 질병 분류 코드는 제거하였습니다(암종이 있기 때문에 불필요한 데이터임). 그리고 변동이 있었던 열들은 재정렬 해주었습니다.

```

[12] # 인덱스 재정렬
df.reset_index(drop=True, inplace=True)

# 데이터 확인
df.info

```

<bound method DataFrame.info of

	발생연도	성별	암종	연령군	발생자수	조발생률
0	1999	남자	간	00-04세	11	0.6
1	1999	남자	고환	00-04세	25	1.4
2	1999	남자	기타 암	00-04세	79	4.5
3	1999	남자	뇌 및 중추신경계	00-04세	45	2.6
4	1999	남자	대장	00-04세	1	0.1
...	...	...	...	...	...	...
14897	2020	여자	자궁체부	85세이상	23	4.0
14898	2020	여자	췌장	85세이상	524	91.9
14899	2020	여자	폐	85세이상	935	163.9
14900	2020	여자	호지킨림프종	85세이상	1	0.2
14901	2020	여자	후두	85세이상	6	1.1

[14902 rows x 6 columns]>

```

# 데이터 타입 확인
df.dtypes

```

```

발생연도    int64
성별        object
암종        object
연령군      object
발생자수    int64
조발생률    float64
dtype: object

```

그리고 제거된 열들로 인해 뒤섞인 index를 reset\_index로 재정렬 해주었으며 dtypes를 통해 발생연도와 조발생률의 데이터 타입이 정상적으로 변경되었음을 확인했습니다.

지금까지 진행한 전처리 내용을 요약해보았습니다.

1. 결측치 처리: '조발생률' 열의 결측치를 제거했습니다.
2. 데이터 필터링: '남녀전체' 성별 데이터와 '발생연도', '연령군', '암종' 열의 통합 데이터를 포함하는 행을 제거했습니다.
3. 데이터 형식 조정: '발생연도'와 '조발생률' 열의 데이터 타입을 적절하게 변환했습니다.
4. 필요한 열 선택 및 재구성: 분석에 필요한 열을 선택하고 필요에 따라 데이터를 재구성했습니다.
5. 데이터 정렬 및 인덱스 재정렬: 데이터를 적절한 순서로 정렬하고 인덱스를 재정렬했습니다.

이제 머신러닝을 위한 데이터 전처리 과정이 끝났습니다. 다음으로 데이터 시각화 과정을 진행해보겠습니다.



# 데이터 시각화

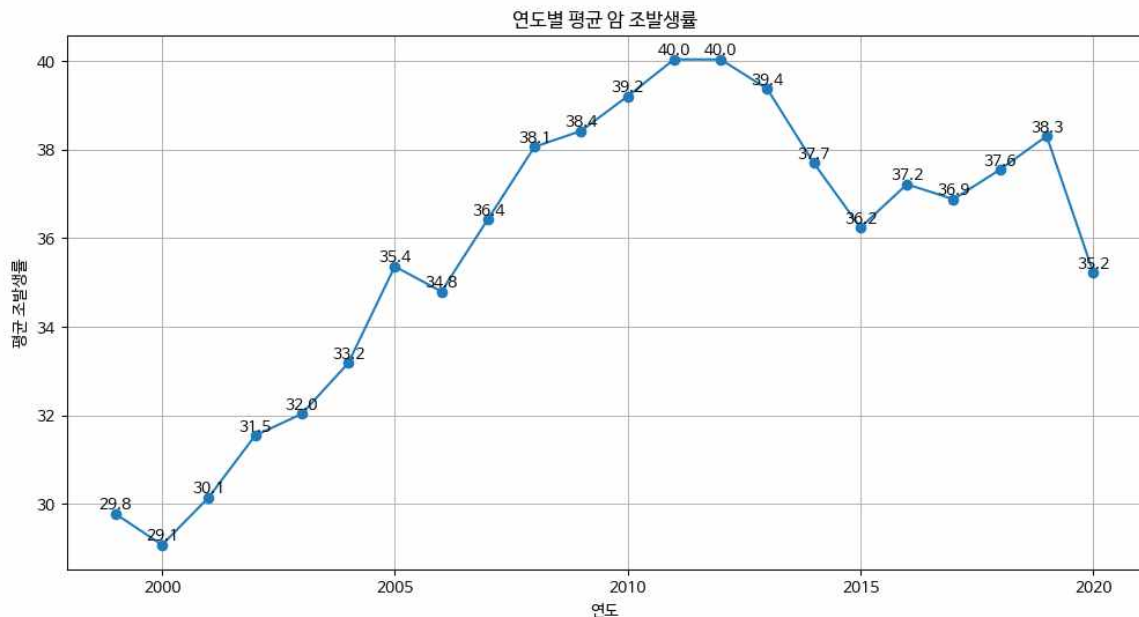
## 1. 연도별 평균 암 조발생률 시각화

먼저 연도별 평균 암 조발생률을 시각화 해보았습니다.

```
[11] import matplotlib.pyplot as plt
      plt.rc('font', family='NanumBarunGothic')

      # '발생연도'와 '조발생률'을 기준으로 평균을 계산
      yearly_avg = df.groupby('발생연도')['조발생률'].mean()

      # 시각화
      plt.figure(figsize=(12, 6))
      plt.plot(yearly_avg, marker='o')
      for year, rate in yearly_avg.items():
          plt.text(year, rate, f'{rate:.1f}', ha='center', va='bottom')
      plt.title('연도별 평균 암 조발생률')
      plt.xlabel('연도')
      plt.ylabel('평균 조발생률')
      plt.grid(True)
      plt.show()
```



해당 그래프는 발생연도와 조발생률 열의 모든 데이터들의 평균을 계산한 것이기에 대한민국 모든 연령대의 국민들의 모든 암 조발생률의 평균일 것입니다. 따라서 해당 그래프는 해당 연도에 전국민 10만 명 중 몇 명의 사람들이 암에 걸렸는지를 확인할 수 있는 그래프입니다. 그래프를 살펴보면 1999년부터 2011년까지 우상향하며 올라가던 암 조발생률이 2012년 이후부터 점차 내려가는 양상을 보이고 있습니다.



## 2. 연도에 따른 암종별 조발생률의 변화 시각화

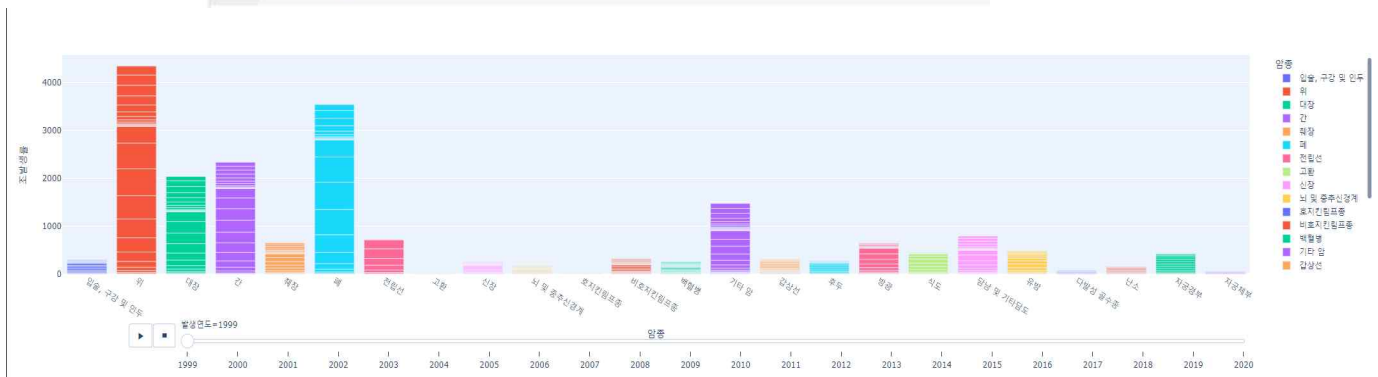
하지만 저는 전체 암의 정보도 궁금하지만 각각 암종 별로 연도에 따른 조발생률의 변화 역시 궁금하기 때문에 해당 내용을 포함한 그래프를 작성해보겠습니다. 하지만 해당 그래프는 한 화면으로 나타내는 것은 어렵기 때문에 애니메이션을 포함한 대화형 그래프로 나타내주는 것이 좋아보입니다. 아래는 해당 애니메이션을 포함한 대화형 그래프를 작성하는 코드입니다. 수업시간에 배웠던 내용을 이용하니 생각보다 어렵지 않게 만들 수 있었습니다.

```
import plotly.express as px

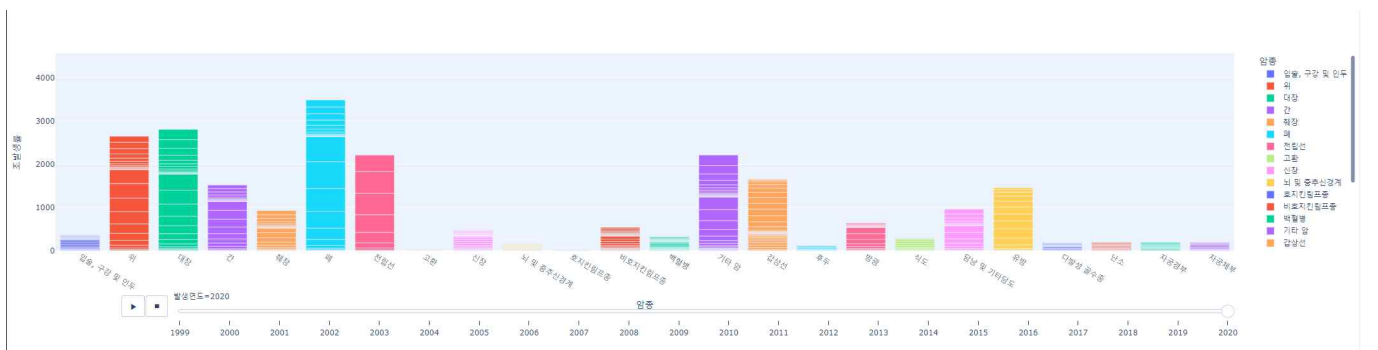
# '모든 암' 카테고리 제거
df = df[df['암종'] != '모든암']

# Plotly를 사용하여 대화형 그래프 생성
fig = px.bar(df, x='암종', y='조발생률', color='암종', animation_frame='발생연도')

# 그래프 표시
fig.show()
```



사진이 잘 보이지는 않지만 대화형 그래프로 좌측 하단의 play 버튼을 누르면 1999년부터 2020년까지 어떤 암들이 많이 발생하고 있는지를 확인할 수 있습니다. 1999년에는 위암이 가장 높은 비율을 차지했음을 알 수 있습니다.



2020년에는 위암 등의 암들보다는 백혈병, 림프종, 등의 호르몬 암들의 수가 많아졌음을 알 수 있습니다.

다음으로 머신러닝을 통한 2025년까지의 암 조발생률 예측을 해보도록 하겠습니다.

# 데이터 분석(머신러닝)

## 1. ADF 테스트

해당 테스트를 통해 데이터가 머신러닝을 하기에 적합한 수준의 시계열 데이터가 맞는지 확인해보고 머신러닝을 진행해보겠습니다.

```
from statsmodels.tsa.stattools import adfuller
```

```
# ADF 테스트 수행
```

```
result = adfuller(yearly_avg)
```

```
# ADF 테스트 결과 출력
```

```
print('ADF 통계량: %f' % result[0])
```

```
print('p-value: %f' % result[1])
```

```
print('임계값:')
```

```
for key, value in result[4].items():  
    print('%t%s: %.3f' % (key, value))
```

```
ADF 통계량: -3.464563
```

```
p-value: 0.008945
```

```
임계값:
```

```
1%: -4.138
```

```
5%: -3.155
```

```
10%: -2.714
```

# ADF 통계량이 5% 임계값보다 작습니다: 이는 모델이 5%의 신뢰 수준에서 데이터가 정상 시계열일 가능성이 높다는 것을 의미합니다.

# p-value가 0.05 (5%)보다 작습니다: 이 역시 데이터가 정상 시계열일 가능성이 높다는 것을 의미합니다.

테스트가 완료되었으니 다음으로 머신러닝을 진행해보겠습니다.

## 2. 머신러닝 및 시각화

머신러닝을 진행해보겠습니다. 머신러닝의 결과를 앞서 만든 연도별 평균 암 조발생률 그래프에 나타내보도록 하겠습니다.

```
from statsmodels.tsa.arima.model import ARIMA
import matplotlib.pyplot as plt
from sklearn.metrics import mean_squared_error
from math import sqrt

# '발생연도'와 '조발생률'을 기준으로 평균을 계산
yearly_avg = df.groupby('발생연도')['조발생률'].mean()

# 훈련 데이터와 테스트 데이터 분리
train = yearly_avg.iloc[:-5]
test = yearly_avg.iloc[-5:]

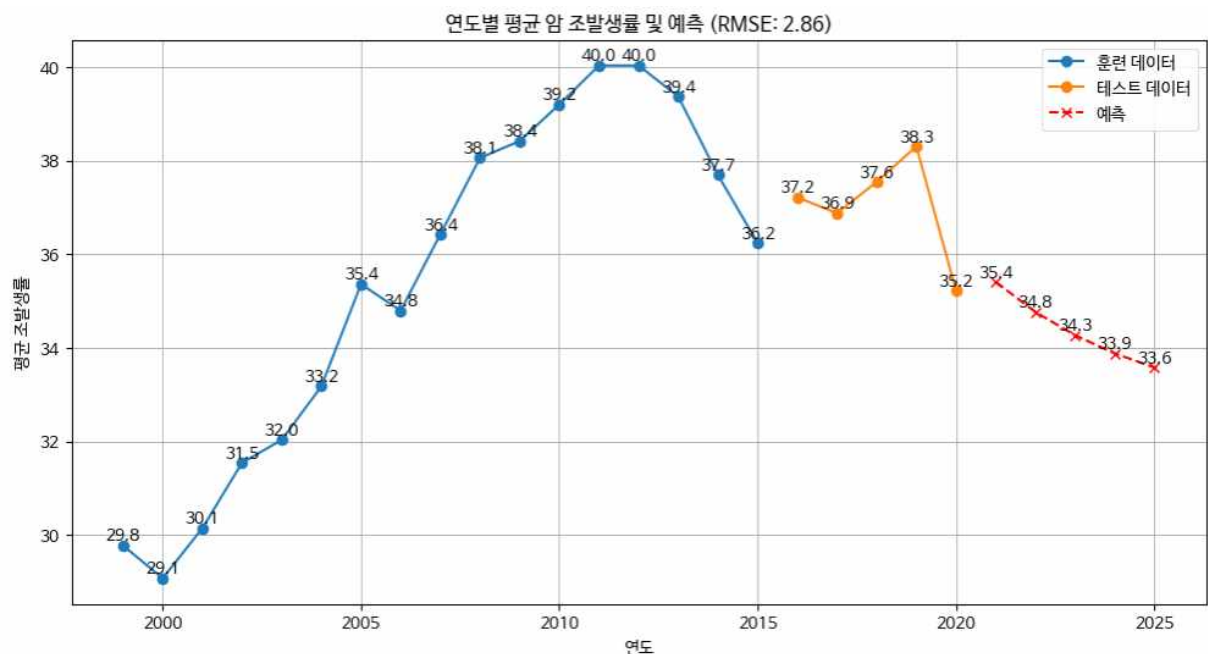
# ARIMA 모델 피팅 및 예측
model = ARIMA(train, order=(1,1,1))
model_fit = model.fit()
forecast = model_fit.forecast(steps=5)

# 성능 평가 (RMSE 계산)
rmse = sqrt(mean_squared_error(test, forecast))

# 예측 결과 시각화
plt.figure(figsize=(12, 6))
plt.plot(train, marker='o', label='Training Data')
plt.plot(test, marker='o', label='Actual Test Data')
plt.plot(range(2021, 2026), forecast, marker='x', linestyle='--', color='red', label='Forecast')

# 각 점 위에 값 표시
for year, rate in train.items():
    plt.text(year, rate, f'{rate:.1f}', ha='center', va='bottom')
for year, rate in test.items():
    plt.text(year, rate, f'{rate:.1f}', ha='center', va='bottom')
for i, rate in enumerate(forecast):
    plt.text(2021 + i, rate, f'{rate:.1f}', ha='center', va='bottom')

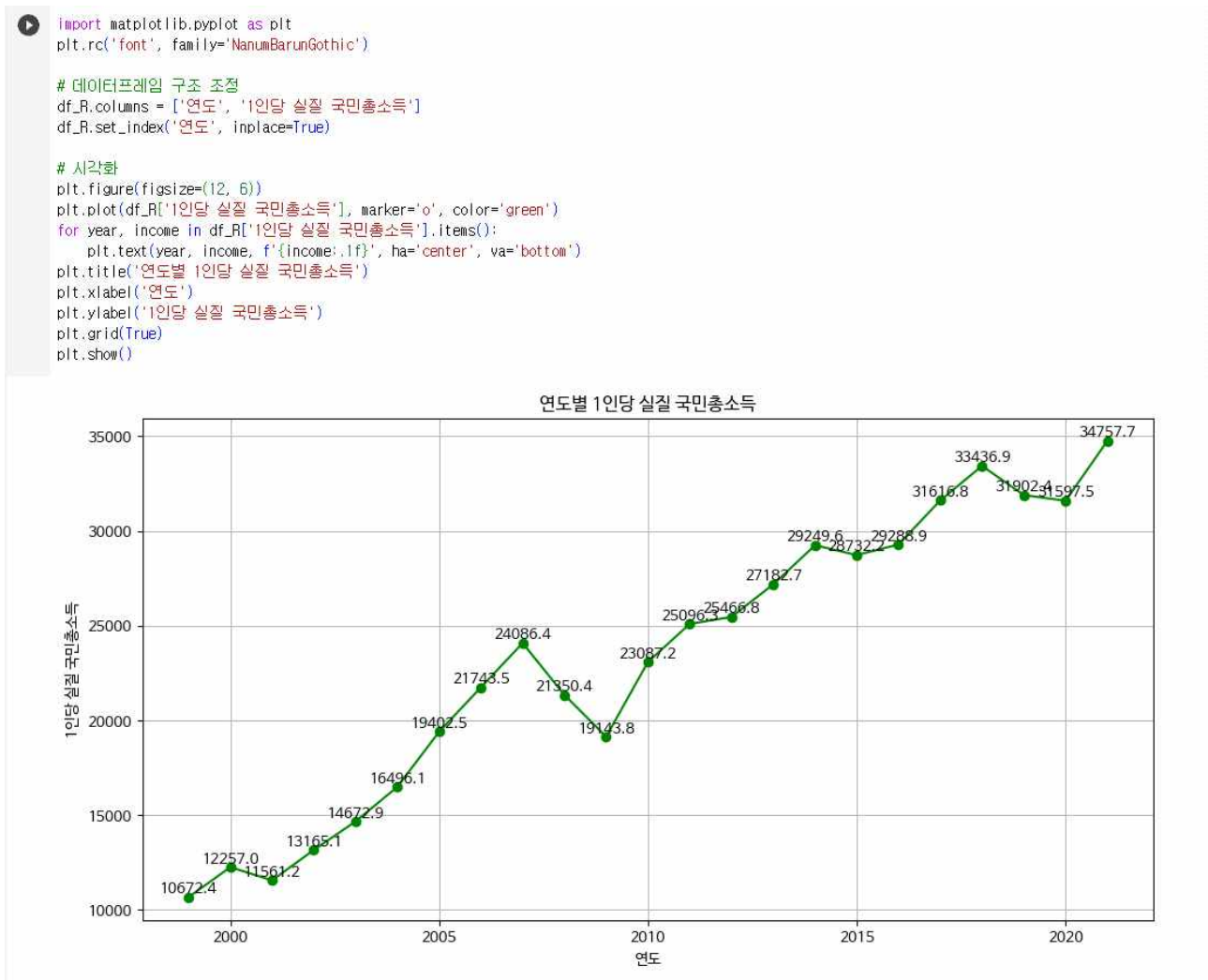
plt.title('연도별 평균 암 조발생률 및 예측 (RMSE: {:.2f})'.format(rmse))
plt.xlabel('연도')
plt.ylabel('평균 조발생률')
plt.legend()
plt.grid(True)
plt.show()
```



머신러닝의 정확도를 확인해보기 위해 RMSE를 Title에 추가해보았습니다. RMSE는 평균제곱오차로 값이 낮을수록 정확하다는 뜻이기에 2.86이면 어느정도 정확하다는 것으로 해석해도 될 것 같습니다.

예측한 내용을 살펴보니 2021년에 살짝 증가하였다가 계속 우하향하는 양상을 보일 것이라고 예측하였습니다.

5단계 분석 결과 및 결론 도출 전 두 번째 데이터인 Real\_Income\_data.csv에 해당하는 그래프를 작성해보겠습니다. 1인당 GDP와 평균 암 조발생률 간의 비교를 위한 그래프입니다.



그래프를 살펴보면 조금의 2007년부터 2009년까지 잠깐의 하락이 존재하였지만(미국발 세계 금융 위기로 인한 것으로 예상) 전체적으로는 우상향하는 모습을 보이고 있습니다.

이제 해당 그래프와 1번 데이터를 이용한 평균 암 조발생률 그래프를 옆에 두고 비교하며 결론을 지어보겠습니다.

# 분석 결과 및 결론

## 1. 그래프 간 비교

```
from statsmodels.tsa.arima.model import ARIMA
import matplotlib.pyplot as plt
from sklearn.metrics import mean_squared_error
from math import sqrt
import pandas as pd

# 'df' 데이터프레임에서 '발생연도'와 '조발생률'을 기준으로 평균을 계산
# 여기서 'df'는 앞서 언급한 암 데이터를 포함하는 데이터프레임입니다.
yearly_avg = df.groupby('발생연도')['조발생률'].mean()

# 훈련 데이터와 테스트 데이터 분리
train = yearly_avg.iloc[:-5]
test = yearly_avg.iloc[-5:]

# ARIMA 모델 피팅 및 예측
model = ARIMA(train, order=(1,1,1))
model_fit = model.fit()
forecast = model_fit.forecast(steps=5)

# 성능 평가 (RMSE 계산)
rmse = sqrt(mean_squared_error(test, forecast))

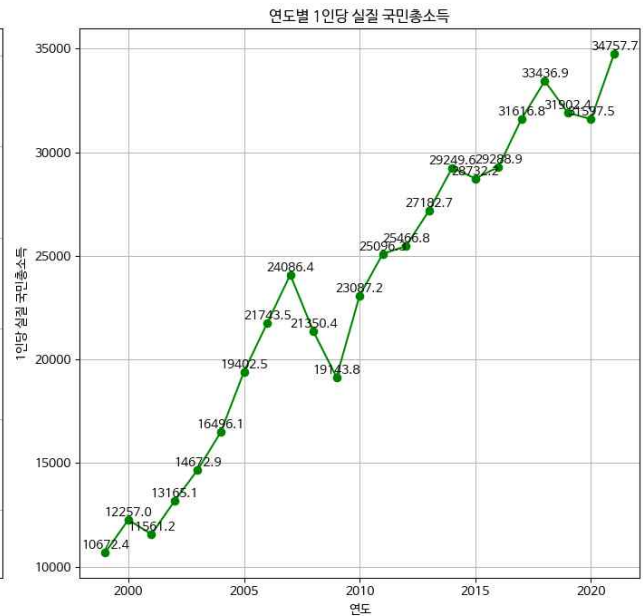
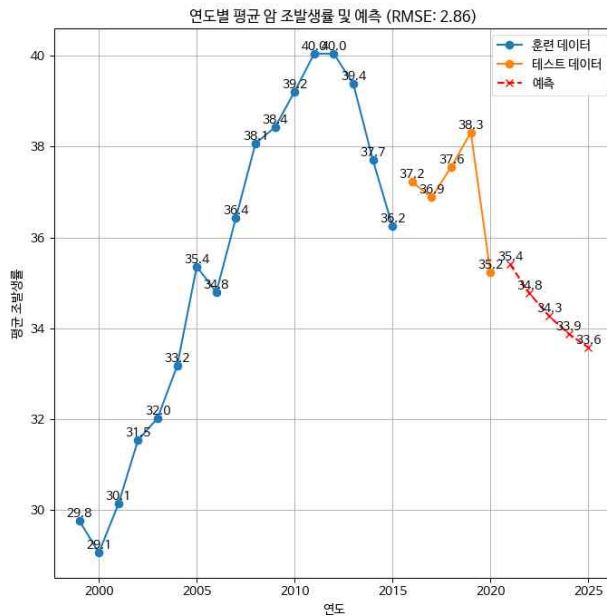
# 비교 그래프 생성
plt.figure(figsize=(14, 7))

# 연도별 평균 암 조발생률 및 예측 시각화
plt.subplot(1, 2, 1)
plt.plot(train, marker='o', label='Training Data')
plt.plot(test, marker='o', label='Actual Test Data')
plt.plot(range(2021, 2026), forecast, marker='x', linestyle='--', color='red', label='Forecast')
for year, rate in train.items():
    plt.text(year, rate, f'{rate:.1f}', ha='center', va='bottom')
for year, rate in test.items():
    plt.text(year, rate, f'{rate:.1f}', ha='center', va='bottom')
for i, rate in enumerate(forecast):
    plt.text(2021 + i, rate, f'{rate:.1f}', ha='center', va='bottom')
plt.title('연도별 평균 암 조발생률 및 예측 (RMSE: {:.2f})'.format(rmse))
plt.xlabel('연도')
plt.ylabel('평균 조발생률')
plt.legend()
plt.grid(True)

# 연도별 1인당 실질 국민총소득 시각화
plt.subplot(1, 2, 2)
plt.plot(df_R['1인당 실질 국민총소득'], marker='o', color='green')
for year, income in df_R['1인당 실질 국민총소득'].items():
    plt.text(year, income, f'{income:.1f}', ha='center', va='bottom')
plt.title('연도별 1인당 실질 국민총소득')
plt.xlabel('연도')
plt.ylabel('1인당 실질 국민총소득')
plt.grid(True)

# 레이아웃 조정
plt.tight_layout()
plt.show()
```

두 그래프를 합쳐서 보여주는 코드입니다. 해당 코드를 실행시켜보면



한 화면에 두 개의 그래프를 동시에 볼 수 있습니다.

처음에 가정하였던 대로 1인당 실질 국민총소득이 증가하면 암 조발생률이 낮아질 것이라는 가정은 옳다고는 할 수 없을 것입니다. 하지만 완전히 틀리지 않은 이유는 2012년 이후부터는 우하향하는 모습을 보이고 있고 머신러닝을 통한 예측 역시 점점 낮아질 것이라고 하기 때문입니다.

## 2. 결론

처음 이번 주제를 선정하였을 때는 연도별 1인당 국민총소득이 계속해서 우상향했다는 사실은 알고 있었지만 암 조발생률의 정보는 모르는 상태에서 아마도 1인당 국민총소득이 늘어남에 따라 암 조발생률이 감소하지 않을까? 라는 가정을 가지고 프로젝트를 진행해보았습니다.

결과론적으로 암 평균 조발생률과 1인당 국민총소득 간의 직접적인 연관성은 찾지 못하였습니다.

하지만 암종에 따른 조발생률의 변화를 그래프를 보면 과거에는 식사 환경이 좋지 못하고 음식의 수준 역시 높지 못하였다보니 위암이 가장 높은 부분을 차지했으나 현재에는 갑상선, 림프종 등등 호르몬 관련 암들이 증가하는 모습을 보입니다. 아마 과거보다 많은 전자파 속에서 살고 있다보니 이러한 암들의 발생률이 높아지지 않았을까? 하는 생각을 해봅니다. 또한 현재는 몸 자체가 힘든 것 보다는 정신적으로 힘든 사람들이 많은 만큼 스트레스가 이러한 결과에 원인이 되지 않았나 하는 생각을 하였습니다.

머신러닝의 예측 결과로보면 앞으로는 암 조발생률이 우하향할 것이라는 예측을 보여줌과 같이 더 이상은 암으로 고통 받는 사람들이 없었으면 하는 바램을 가지고 프로젝트를 마무리해보고자합니다.