5주차

수업 내용

- Kaggle Dataset를 불러오는 방법을 알 수 있다.
- KNN의 장단점을 확인해 볼 수 있다.
- 선형회귀를 이용하여 회귀 문제를 풀 수 있다.
- 선형회귀를 활용한 로지스틱 회귀를 다뤄볼 수 있다.

목차

대한민국 미래 인구수 예측해보기

리가 수업 진행하기 앞서 전 시간에 분류와 관련된 내용으로 수업을 진행했다. 우리가 어떠한 문제를 기반으로 회귀 문제를 해결 해보자.

population world since 1960 to 2021

World Bank Data of World Population By Year

https://www.kaggle.com/datasets/fredericksalazar/population-world-since-1960-to-2021

이 사이트로 들어가면 Kaggle 이라는 사이트로 들어가진다.

Kaggle이 무엇인지 간단하게 설명을 해보자면

> AI 경진대회, 데이터 셋 등 인공지능 관련된 사이트이다.

위 링크에서는 각 국가별로 인구수 데이터를 제공하고 있다.

어떤 방식으로 문제 해결 해볼까?

우리가 해결 해볼 문제는

대한민국 데이터를 뽑고,

1960 ~ 2021년 데이터 기반 학습하고 앞으로 있을 다음 년도(2022)년 것을 한번 예측해보자.

그러한 문제를 해결해보면서

우리가 문제 해결할 문제가 전 시간에 배운 KNN이 적합한지 확인해보고,

KNN보다 더 우수한 기법을 탐색해서 적용을 해보자.

우리는 KNN 기법을 사용하기 전에 우리가 데이터를 불러와야 한다. 위 링크를 눌러서 실습을 해보자.

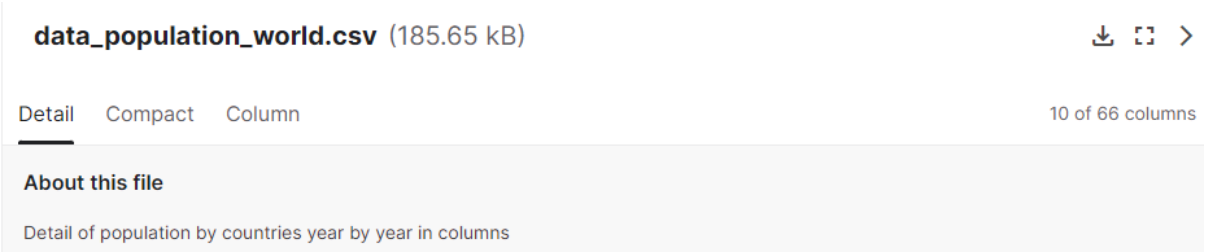
어떠한 데이터를 사용할 것인가?

우리가 데이터를 불러올 필요가 있다.

우리가 위에 있는 링크를 접근하고, DataSet를 받으면 여러개의 파일들이 존재한다.

각 csv에 따라서 데이터를 제공하는 종목이 다르다 그렇기에 데이터를 어떤 것을 사용해야할지 어느정도 확인할 필요가 있다.

그러한 데이터를 잘 아는건, DataSet를 만든 사람이 잘 안다. 그렇기에 Kaggle 링크를 잘 확인한다.



data_population_world.csv 영역에 Detail 영역을 잘 확인하자.

그 내용을 해석해보니 “연도, 국가별로 인구 세부 정보”라는 내용이다.

그러면 국가에 대한 데이터가 존재할 것임을 확인 할 수 있다.

그 다음으로

metadata_contries.csv를 확인해본다.

작성자가 작성한 기준으로는 이 데이터가 무엇을 의미하는지 자세히 나와있진 않다.

metadata_countries.csv (19.86 kB)

Detail Compact Column

Country Name	Country Code	Region	Income_Group
265 unique values	266 unique values	[null] 49% África al sur del Sa... 18% Other (89) 33%	Ingreso alto 30% Países de ingreso ... 20% Other (132) 50%
Aruba	ABW		Ingreso alto
	AFE		Agregados
Afganistán	AFG	Asia meridional	Países de ingreso bajo
	AFW		Agregados
Angola	AGO	África al sur del Sahara (excluido altos ingresos)	Países de ingreso mediano bajo
Albania	ALB	Europa y Asia central (excluido altos ingresos)	Ingreso mediano alto
Andorra	AND		Ingreso alto
El mundo árabe	ARB		Agregados
Emiratos Árabes Unidos	ARE		Ingreso alto
Argentina	ARG	América Latina y el Caribe (excluido altos ingresos)	Ingreso mediano alto
Armenia	ARM	Europa y Asia central (excluido altos ingresos)	Ingreso mediano alto

대충 눈으로 확인 해봤을 때, 어떠한 국가를 기준으로 가져온건지에 관련된 내용을 의미하는 것 같다.

결국 우리가 사용해야할 것은 “data_popularion_word.csv”를 불러와서 사용하면 된다라는 것을 확인해볼 수 있다.

데이터를 불러오자.

데이터 불러오는 방법은 다양하지만, 쉽게 불러오는 방법은 우리가 전시간부터 사용했던 Pandas 라이브러리를 사용하면 된다.

데이터를 불러오는 방법은 다음과 같다.

```
import pandas as pd
```

```
df = pd.read_csv('data_population_world.csv')
```

데이터를 불러오면 다음과 같이 나올 것이다.

	Country Name	Country Code	Indicator Name	Indicator Code	1960	1961	1962	1963	1964	1965	...
0	Aruba	ABW	Población, total	SP.POP.TOTL	54608.0	55811.0	56682.0	57475.0	58178.0	58782.0	...
1	NaN	AFE	Población, total	SP.POP.TOTL	130692579.0	134169237.0	137835590.0	141630546.0	145605995.0	149742351.0	...
2	Afganistán	AFG	Población, total	SP.POP.TOTL	8622466.0	8790140.0	8969047.0	9157465.0	9355514.0	9565147.0	...
3	NaN	AFW	Población, total	SP.POP.TOTL	97256290.0	99314028.0	101445032.0	103667517.0	105959979.0	108336203.0	...
4	Angola	AGO	Población, total	SP.POP.TOTL	5357195.0	5441333.0	5521400.0	5599827.0	5673199.0	5736582.0	...
...	...	...	...	...	...	...	...	...	...	...	...
261	Kosovo	XKX	Población, total	SP.POP.TOTL	947000.0	966000.0	994000.0	1022000.0	1050000.0	1078000.0	...
262	Yemen, Rep. del	YEM	Población, total	SP.POP.TOTL	5542459.0	5646668.0	5753386.0	5860197.0	5973803.0	6097298.0	...
263	Sudáfrica	ZAF	Población, total	SP.POP.TOTL	16520441.0	16989464.0	17503133.0	18042215.0	18603097.0	19187194.0	...
264	Zambia	ZMB	Población, total	SP.POP.TOTL	3119430.0	3219451.0	3323427.0	3431381.0	3542764.0	3658024.0	...
265	Zimbabwe	ZWE	Población, total	SP.POP.TOTL	3806310.0	3925952.0	4049778.0	4177931.0	4310332.0	4447149.0	...

데이터 전처리하기

일단 우리가 봐야할 항목은 우리나라가 존재하는지를 확인해야한다.

그 전에 데이터 구조를 먼저 확인해본다.

데이터 구조 확인하는 방법은 pandas에서 제공하고 있기에 pandas docs 에 검색해본다.

pandas에서 info 메소드로 확인을 해볼 수 있다고 알려져 있다.

그렇기에 우리가 다음과 같이 입력해본다.

```
df.info()
```

```
RangeIndex: 266 entries, 0 to 265
Data columns (total 67 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Country Name    264 non-null    object
 1   Country Code    266 non-null    object
 2   Indicator Name  266 non-null    object
 3   Indicator Code  266 non-null    object
 4   1960            264 non-null    float64
 5   1961            264 non-null    float64
 6   1962            264 non-null    float64
 7   1963            264 non-null    float64
...
 58  2014            265 non-null    float64
 59  2015            265 non-null    float64
 60  2016            265 non-null    float64
 61  2017            265 non-null    float64
 62  2018            265 non-null    float64
 63  2019            265 non-null    float64
 64  2020            265 non-null    float64
 65  2021            265 non-null    float64
 66  Unnamed: 66     0 non-null      float64
dtypes: float64(63), object(4)
memory usage: 139.4+ KB
```

(뭔가 많은 것들이 출력되는건데 암튼 데이터 영역이 좀 큰거다.)

우리가 봐야하는 것은

- 전체 행의 개수
- 각 열에 대한 non-null 개수

보니깐 Country Name은 **null 개수가 2개 나오는 것을 확인** 해볼 수 있다.

우리가 검색할 땐 나중에 하겠지만, **null로 인해서 한국 국가만 추출하기엔 어려운** 것 같다.

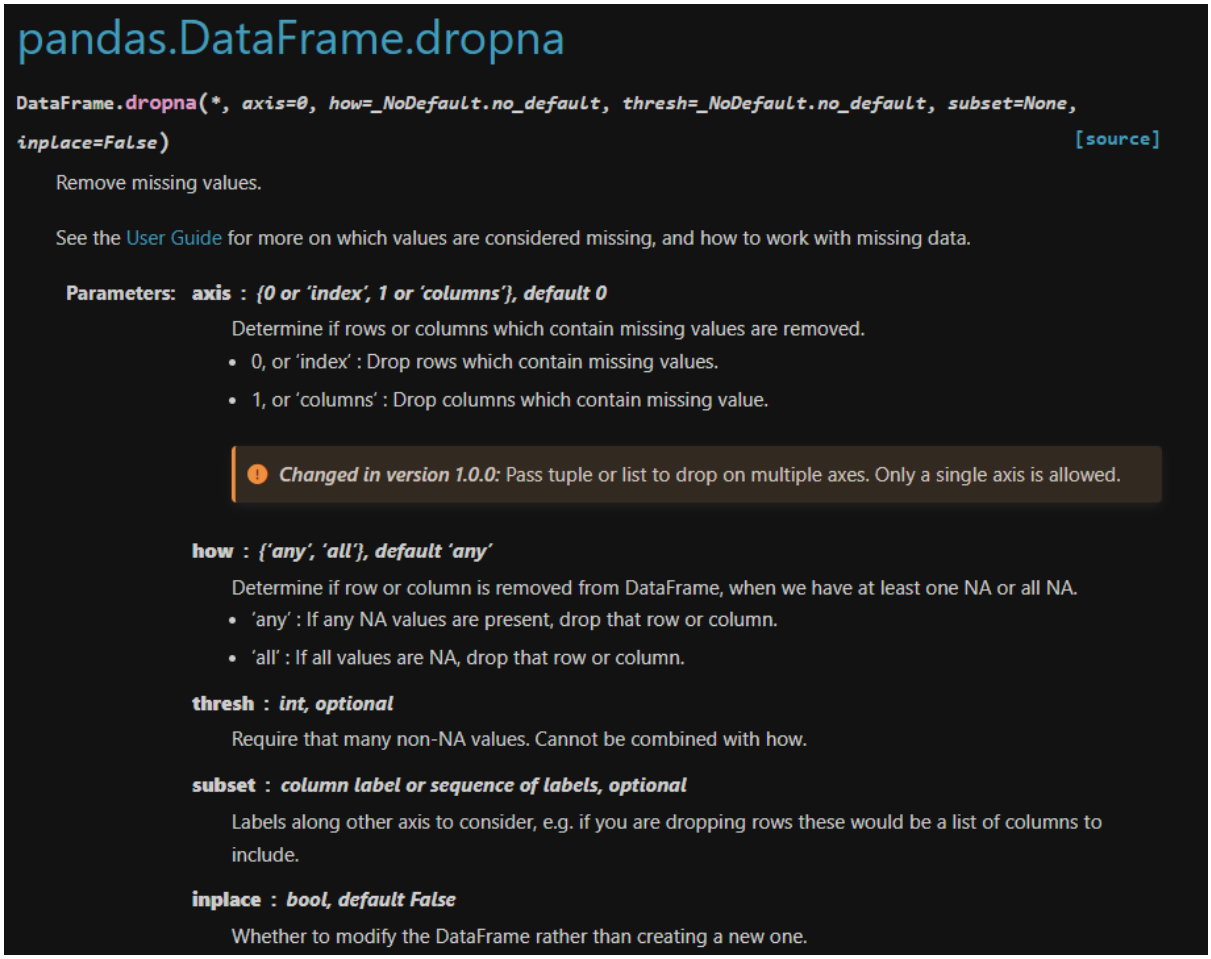
그렇기에 Country Name 영역에 null를 지우는 처리가 필요하다

그리고 나중에 말이지만, 우리가 원하는 년도별 인구수 제외한 나머지 열을 마지막에 지울 필요가 있다.
그러한 것은 우리가 한국에 대한 데이터를 구했을 때 적용하면 될 것 같다.

그리고 이상한 데이터가 있다.
맨 마지막 열에 “Unnamed: 66”이 있는데, 지워줄 필요가 있어 보인다.

null인 영역 제거하기

우리는 “Country Name” 영역에 Nan 있으면 지우는 코드를 작성한다.



pandas.DataFrame.dropna

DataFrame.dropna(*, axis=0, how=_NoDefault.no_default, thresh=_NoDefault.no_default, subset=None, inplace=False) [source]

Remove missing values.

See the User Guide for more on which values are considered missing, and how to work with missing data.

Parameters:

axis : {0 or 'index', 1 or 'columns'}, default 0

Determine if rows or columns which contain missing values are removed.

- 0, or 'index' : Drop rows which contain missing values.
- 1, or 'columns' : Drop columns which contain missing value.

Changed in version 1.0.0: Pass tuple or list to drop on multiple axes. Only a single axis is allowed.

how : {'any', 'all'}, default 'any'

Determine if row or column is removed from DataFrame, when we have at least one NA or all NA.

- 'any' : If any NA values are present, drop that row or column.
- 'all' : If all values are NA, drop that row or column.

thresh : int, optional

Require that many non-NA values. Cannot be combined with how.

subset : column label or sequence of labels, optional

Labels along other axis to consider, e.g. if you are dropping rows these would be a list of columns to include.

inplace : bool, default False

Whether to modify the DataFrame rather than creating a new one.

지우는 코드는 다음과 같다.

우리가 핵심적으로 볼 것은 subset이다.
subset은 우리가 nan 지우는 대상을 의미한다.
그러니 우리는 “Country Name” 영역을 넣어서 Country Name이 Nan 인 요소를 제거한다.

그러면 다음과 같이 넣어서 처리를 하면 된다.

```
df = df.dropna(subset='Country Name')
```

df 출력해보면 알 수 있다시피 눈으로 보이는 Nan은 사라진걸 확인할 수 있다.

	Country Name	Country Code	Indicator Name	Indicator Code	1960	1961	1962	1963	1964	1965	...
0	Aruba	ABW	Población, total	SP.POP.TOTL	54608.0	55811.0	56682.0	57475.0	58178.0	58782.0	...
2	Afganistán	AFG	Población, total	SP.POP.TOTL	8622466.0	8790140.0	8969047.0	9157465.0	9355514.0	9565147.0	...
4	Angola	AGO	Población, total	SP.POP.TOTL	5357195.0	5441333.0	5521400.0	5599827.0	5673199.0	5736582.0	...
5	Albania	ALB	Población, total	SP.POP.TOTL	1608800.0	1659800.0	1711319.0	1762621.0	1814135.0	1864791.0	...
6	Andorra	AND	Población, total	SP.POP.TOTL	9443.0	10216.0	11014.0	11839.0	12690.0	13563.0	...
...	...	...	...	...	...	...	...	...	...	...	...
261	Kosovo	XKX	Población, total	SP.POP.TOTL	947000.0	966000.0	994000.0	1022000.0	1050000.0	1078000.0	...

한국인 데이터만 가져오기

한국 데이터만 가져오는 방법은 데이터를 눈으로 확인 했을 때,

- Country Name
- Country Code

이 한국인 것만 가져오면 된다.

그러기에 방법이 이 교재에서는 2가지를 알려주겠다.

Country Name으로 한국 찾아보기

Country Name 기준으로 찾는 방법은 Korea 포함이 되어 있는 것을 찾으면 된다.

우리가 앞서 써왔던 기능을 활용하면 다음처럼 쓸 수 있다.

```
ze
```

```
df[df['Country Name'].str.contains("korea")]
```

✓ 0.0s

	Country Name	Country Code	Indicator Name

0 rows × 67 columns

그러나 결과가 없는 것을 확인해 볼 수 있다.

왜 없는지를 한번 탐구해봐라

왜 없는 이유?

한번 데이터 내용물들을 확인해보자.

	Country Name	Country Code	Indicator Name	Indicator Code	1960	1961	1962	1963	1964	1965
126	Corea, República de	KOR	Población, total	SP.POP.TOTL	25012374.0	25765673.0	26513030.0	27261747.0	27984155.0	28704674.0

Country Name이 한국 기준으로는 “Corea, República de” 이렇게 기록이 되어 있음을 확인해볼 수 있다.

Country Name으로 찾고 싶다면 다음처럼 쓰면 된다.

```
df[df['Country Name'].str.contains("Corea")]
```

	Country Name	Country Code	Indicator Name	Indicator Code	1960	1961	1962	1963	1964	1965
126	Corea, República de	KOR	Población, total	SP.POP.TOTL	25012374.0	25765673.0	26513030.0	27261747.0	27984155.0	28704674.0
193	Corea, República Popular Democrática de	PRK	Población, total	SP.POP.TOTL	11655666.0	11916515.0	12180505.0	12475236.0	12787523.0	13105946.0

2 rows × 67 columns

(~~어째서 북한 데이터를 구한거지? ㅎㄷㄷ~~)

남한이랑 북한은 국가 명이 Corea가 포함되어 있다. 그렇기에 한국 데이터로 구하고 싶으면 다음처럼 쓰면 되겠다.

```
df[df['Country Name'].str.contains("Corea, República de")]
```

내용을 쓰는데 번거롭다. 한번 Country Code로 검색해보는게 확실하게 깔끔하게 구할 수 있을 것 같다.

Country Code로 한국 찾기

눈치가 빠른 학생이라면 국가 코드가 뭔지를 알 것이다. 앞에서 데이터 봤다시피 “KOR”를 의미한다.

뭐 잘 모르겠다면 한번 구글링를 해보라

나라 이름	숫자	alpha-3
대한민국	410	**KOR**
덴마크	208	DNK
도미니카 공화국	214	DOM
도미니카 연방	212	DMA
113행 더 보기		

암튼 “KOR”이다

그러면 우리는 Country Code가 KOR인 것만 가져오면 된다.
다음처럼 코드를 작성해볼 수 있다.

```
df[df['Country Code']=='KOR']
```

	Country Name	Country Code	Indicator Name	Indicator Code	1960	1961	1962	1963	1964	1965	...
126	Corea, República de	KOR	Población, total	SP.POP.TOTL	25012374.0	25765673.0	26513030.0	27261747.0	27984155.0	28704674.0	...

그럼 깔끔하게 나온다.

우리는 최종적으로 Country Code 기준으로 한국 데이터를 불러오고, 그 데이터를 df에 저장하자.

```
df = df[df['Country Code']=='KOR']
```

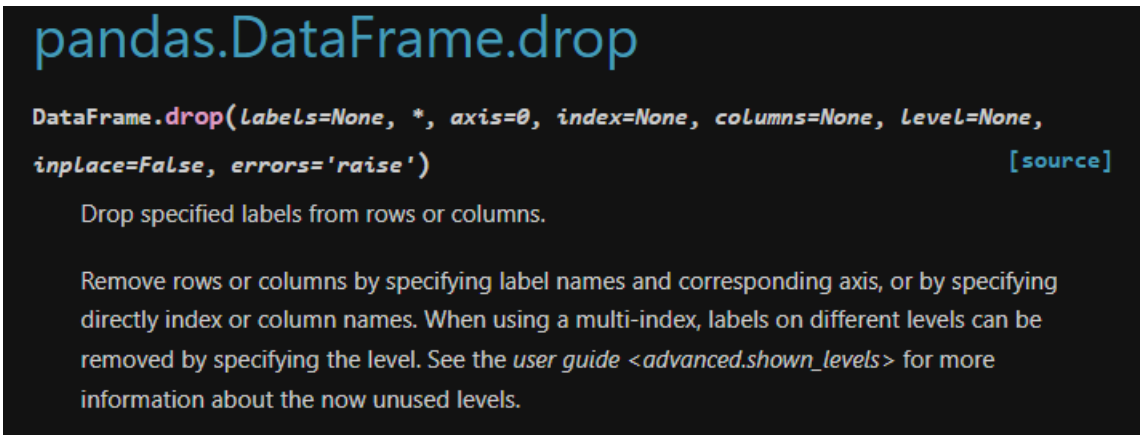
연도별 인구수 제외한 열 삭제

이제 우리가 원하는건 연도별 인구수만 가져오면 된다.

연도만 추출하는 것보단 불필요한 열을 삭제하는 것이 코드를 짧게 짤 수 있을 것 같다.
우리가 불필요한 것은 다음 이미지영역이라고 보면 된다.

Country Name	Country Code	Indicator Name	Indicator Code
Corea, República de	KOR	Población, total	SP.POP.TOTL

열을 지우는 방법은 전 시간에서 사용한 코드를 활용해보자.



pandas에서 이미 제공하고 있다.
그 코드를 활용하면 다음과 같다.

```
df = df.drop(columns = ['Country Name','Country Code','Indicator Name','Indicator Code','Unnamed: 66'])
```

	1960	1961	1962	1963	1964	1965	1966	1967	1968	1969
126	25012374.0	25765673.0	26513030.0	27261747.0	27984155.0	28704674.0	29435571.0	30130983.0	30838302.0	31544266.0

불필한 데이터가 없어진 것을 확인할 수 있다.

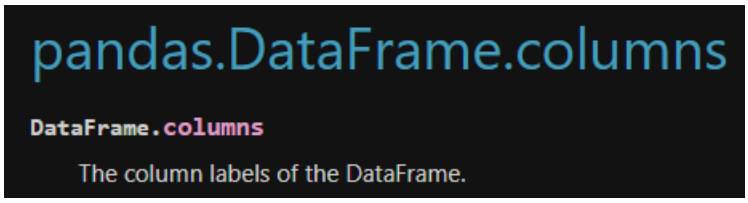
numpy로 변환하기

이제 우리가 데이터를 본격적으로 사용하기 앞서 numpy로 변환할 필요가 있다.

우리가 X 데이터는 “년도” 즉 1960~2021 범위를 가지는 list가 필요하다.
그 다음 Y 데이터는 우리가 아까 구한 연도별 인구수를 추출한다.

일단 X 데이터는 우리가 dataframe에 columns 가져오면 해결이 된다.

하는 방법은 지난시간에 언급했던 df.columns 변수를 가져오는 것이다.



다음처럼 코드를 작성해볼 수 있다.

```
X = df.columns.to_numpy(dtype='int')
```

X를 출력하면 다음처럼 된다.

```
array([1960, 1961, 1962, 1963, 1964, 1965, 1966, 1967, 1968, 1969, 1970,
       1971, 1972, 1973, 1974, 1975, 1976, 1977, 1978, 1979, 1980, 1981,
       1982, 1983, 1984, 1985, 1986, 1987, 1988, 1989, 1990, 1991, 1992,
       1993, 1994, 1995, 1996, 1997, 1998, 1999, 2000, 2001, 2002, 2003,
```

```
       2004, 2005, 2006, 2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014,
       2015, 2016, 2017, 2018, 2019, 2020, 2021])
```

워 대체재로 np.arange(1960,2022) 쓰면 동일한 결과가 나오니 참고만 하자

이제 Y 데이터는 dataframe에 있는 데이터를 numpy 배열을 변환하면 된다.

```
Y = df.to_numpy()
```

```
Y = Y.reshape((-1,1))
```

여기에서 reshape는 '차원을 바꾸다' 라는 의미를 가진다. 그렇게 하는 이유는 앞으로 우리가 KNN 실습을 진행할 때, 필요하기 때문이다.

출력하면 우리나라 연도별 인구수를 확인할 수 있다.

```
array([[25012374.],
       [25765673.],
       [26513030.],
       [27261747.],
       [27984155.],
...
       [51361911.],
       [51585058.],
       [51764822.],
       [51836239.],
       [51744876.]])
```

일반적인 리스트와 다르게, 2차원처럼 되어있다. 그렇게 한 이유가, 우리가 학습을 할 때 무슨 데이터로 학습해야하는지 뚜렷하게 묶어야 하기 때문이다.

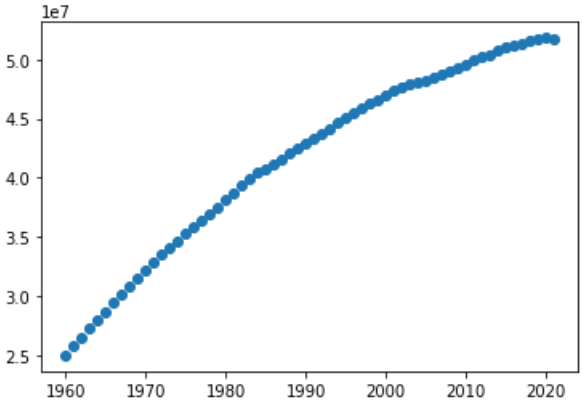
이제 데이터를 구해봤으니 한번 그래프를 그려보자.

데이터 시각화 해보기

matplotlib 라이브러리를 활용하면 그래프를 쉽게 그릴 수 있다.

```
import matplotlib.pyplot as plt

plt.scatter(X,Y)
```



1960년때 증가하다가 마지막 2020년에 줄어든 것을 확인 해볼 수 있다. 우리가 4주차에서 했었을 때 자연인구가 줄어든다는 증명을 더한 셈이다.

인구수 예측하는 KNN 머신러닝 만들기

이제 5주차에서 배운 KNN 기법을 이용해서 인구수 예측하는 머신러닝을 만들어 볼까 한다.

우리 해야하는 문제는 숫자를 예측해야하는 문제이므로, 회귀 문제라고 볼 수 있다.

전 시간에는 전공, 문학-시 분류하는 문제이기에 조금 다르다.

KNN 회귀를 불러와서 불러와보자.

KNN 회귀는 당연히 sklearn 에서 지원한다.

sklearn.neighbors.KNeighborsRegressor

class sklearn.neighbors.**KNeighborsRegressor**(*n_neighbors=5, *, weights='uniform', algorithm='auto', leaf_size=30, p=2, metric='minkowski', metric_params=None, n_jobs=None*) [source]

Regression based on k-nearest neighbors.

The target is predicted by local interpolation of the targets associated of the nearest neighbors in the training set.

Read more in the User Guide.

New in version 0.9.

Parameters: **n_neighbors : *int, default=5***

Number of neighbors to use by default for `kneighbors` queries.

KNeighborsRegressor 부분은

sklearn.neighbors 안에 포함되어 있는 것이다. 그러면 다음처럼 import 하자

```
from sklearn.neighbors import KNeighborsRegressor
```

다음처럼 객체를 생성한다. 여기에서 n를 얼마만큼 쓸것인지 적혀져 있는데, 이 교재에서는 5로 설정할 것이다.

그리고 연산 속도를 높히기 위해 n_jobs를 -1로 둔다.

```
model = KNeighborsRegressor(n_neighbors= 5,n_jobs=-1)
```

객체 생성 후 학습을 진행해야한다. 우리는 전에서 구한 X,Y 변수를 통해 학습을 시켜 준다.

```
model.fit(X,Y)
```

학습이 완료가 되었으면, 이제 predict를 해보자.

그러면 우리는 어떤식으로 predict 할까?

KNN은 가장 가까운 이웃 기준이다.

우리는 X 데이터 기준 즉, 예측하고자 하는 년도가 가장 가까운 X 데이터가 가지고 있는 Y 데이터의 평균으로 예측이 된다.

그러면 2050년 인구수를 예측해볼까?

```
model.predict([[2050]])
```

다음처럼 입력하면 아래처럼 결과가 출력된 것을 확인할 수 있다.

```
array([[51658581.2]])
```

51658551 명이라는 내용이 나왔다.

1960년 ~ 2020년 데이터 중에서 2050과 가장 가까운 년도가 무엇일까?

당연히 2021, 2020, 2019,2018,2017년 이다. 그러면

예측할 때

$$pred = (2017년인구수 + 2018년인구수 + 2019년인구수 + 2020년인구수 + 2021년인구수)/5$$

실제랑 똑같은지 확인해보자.

우리가 학습한 데이터를 확인해본다. 아래처럼 나오는데, 그러면 다음처럼 수식을 적용해보면 될 것이다.

```
np.mean(Y[-5:])
```

여기에서 Y[-5]는 1960~2021년 인구수 데이터 중에 가장 최근데이터 즉 2017~2021년 인구수를 추출하는 것이고, np.mean은 말그대로 평균을 구해주는 친구이다.

이 코드를 실행하면 위에서 predict 했던 결과 값이 똑같이 나오는 것을 확인해 볼 수 있다.

```
51658581.2
```

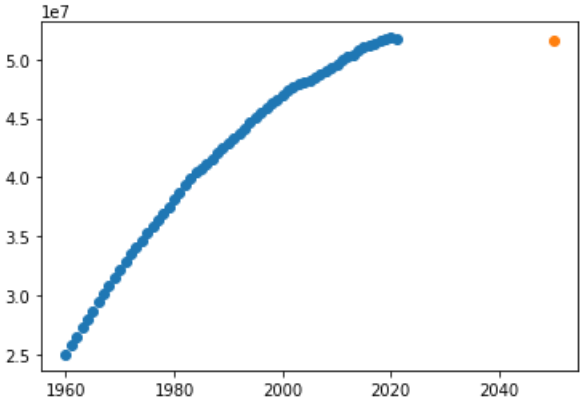
그러면 2100년 인구수를 KNN 기반으로 예측하면 어떻게 될까? 잘 생각해보자.

정답 말하면, 2050년 인구수 예측값과 똑같다. 왜냐하면 학습한 데이터가 1960~2021년 밖에 없기 때문이기에 똑같이 나온다.

그러면 그래프를 띄움으로써 확인해보자.

```
import matplotlib.pyplot as plt

plt.scatter(X,Y)
plt.scatter([2050],model.predict([[2050]]))
```



다음처럼 그래프가 출력 될것이다. 다른 방법 머신러닝 기법을 공부해보자.

LinearRegression 작동 방식

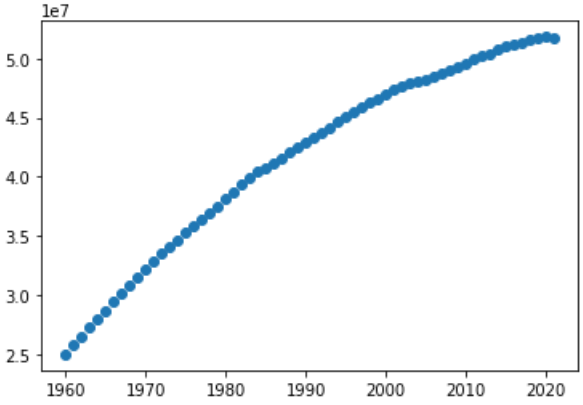
LinearRegression은 선형회귀라는 내용이다.

그 내용은 어느 하나의 데이터를 하나의 일차 방정식으로 만들어 주는 방법이다.

일단 다음과 같은 데이터가 있다고 치자.

이 수 많은 데이터를 하나의 선분으로 표현을 해본다고 생각해보자.

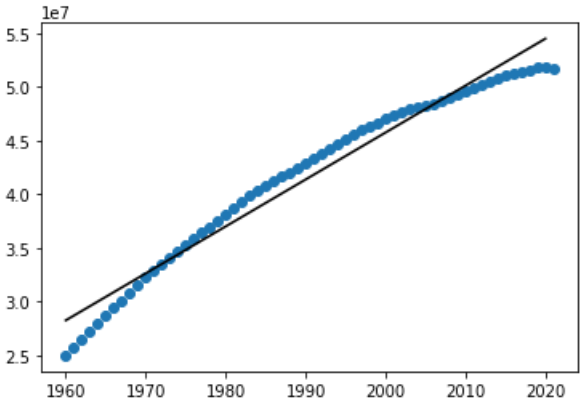
선분을 그린다면 다음과 같다.



쉽게 말하면 수 많은 데이터에 적합한 일차 방정식을 구해주는 거나 마찬가지다.

그러면 여기에서 쓰인 함수는 어떤 것일까?

$$y = ax + b$$



중학생 때 많이 보던 것일 텐데, 여기에서 a(가중치)와 b(절편) 을 선형회귀에서 그 값들을 구한다라고 보면 된다.

그러면 KNN와 선형회귀는 무슨 차이인가?

여기에서 KNN와 차이점은 KNN은 주변에 있는 데이터 기반로 예측하지만, 선형회귀는 각 데이터를 일차 방정식처럼 구해서 예측하는 것이다.

우리가 앞으로 미래 인구수를 측정할 때 KNN보단 선형이라든지, 비선형 같은 것으로 해야 인구를 예측할 때 좀 더 자연 스러울 것이다.

LinearRegression으로 인구수 예측하기

```
from sklearn.linear_model import LinearRegression
```

LinearRegression 클래스를 활용하여 KNN 회귀 문제를 쉽게 해결할 수 있다. 사용하는 코드는 기존에 사용하던 sklearn과 유사하다.

다음처럼 객체를 생성하자.

```
model1 = LinearRegression()
```

생성한 후에 우리는 학습을 시킬 필요가 있다. fit 메소드로 사용해서 학습하면 된다.

```
model1.fit(X,Y)
```

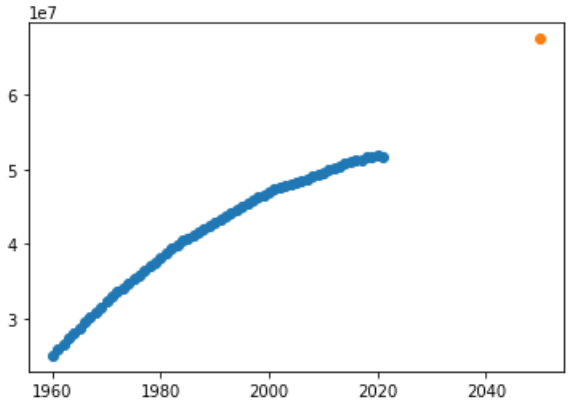
학습하고나서 2050년 인구수를 예측해보자.

```
model1.predict([[2050]])
```

이것을 치면 2050년이 예츠기 되는데, KNN 예측 결과값보다 높게 나오는 것을 확인해볼 수 있다.

그래프로 한번 출력해볼까?

```
plt.scatter(X,Y)
plt.scatter([2050],model1.predict([[2050]]))
```



다음처럼 걀과값이 나오는 것을 확인 해볼 수 있다.

학습한 데이터가 년도가 높아질수록 결과값이 높아지기에 미래 인구수 예측할 때 현재, 과거 인구수보다 더 큰 인구수를 예측한 것이나 마찬가지다.

원리는 아까도 말했다시피 y = ax+b 식에서 가중치(a), 절편(b)를 구한것이다.

그러면 가중치와 절편은 어디에 알아낼 수 있을까?

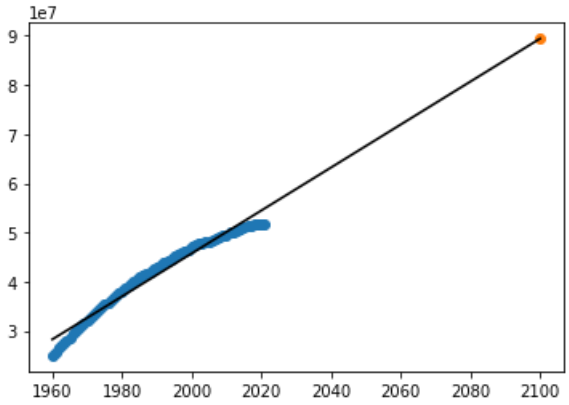
다음처럼 입력하여 확인해보자.

```
model1.coef_, model1.intercept_
```

왼쪽부터 가중치, 절편 정보들이 들어간 영역이다.

그러면 가중치와 절편을 이용해서 선형회귀 선을 같이 그려보자.

```
plt.scatter(X,Y)
plt.scatter([2100],model1.predict([[2100]]))
plt.plot(np.arange(1960,2101),(model1.coef_*np.arange(1960,2101)+model1.intercept_).reshape((-1,)),c='black')
```



입력한 값인 년도를 x 에 대입하고 나온 값 y로 예측이 되는 것을 확인 해볼 수 있다.
그러면 앞에서 선형회귀는 다음처럼 식이 나오는 것을 확인해볼 수 있다.

$$y = 436690.84344388 * 년도수 + -8.2764356e + 08$$

누가 더 성능이 우수할까?

여기에서 성능이 우수하다는 요인인 정확도로 비교해보면 된다.
sklearn에서는 각 모델을 평가할 수 있는 score 메소드가 있다.
우리가 학습한 데이터인 X,Y 데이터 기반으로 정확도를 확인해보자.

여기 교재에서는 model은 KNN, model1은 선형회귀이다.

```
model.score(X,Y)
```

```
0.9992611271228364
```

```
model1.score(X,Y)
```

```
0.9659555460566848
```

위 수치상으로는 선형회귀보다 KNN이 성능이 우수하는 것을 확인해볼 수 있다.

과연 KNN이 우수할까?

우리는 실전에서도 사용이 가능한 모델이 필요하다. KNN은 학습한 데이터 범위가 넘어가는 순간 예측을 못하는 방면, 선형회귀는 학습한 데이터의 범위가 넘어가더라도 데이터 특성에 맞춰 예측을 해주는 편이다.

한마디로 말해서, 학습한 데이터에 따라서 알맞는 머신러닝 기법이 조금씩 달라진다. 그렇기에 우리가 머신러닝 기법을 다양하게 배우는 것이다.