# AI 특강

**Prof. Jibum Kim**

**Department of Computer Science & Engineering**

**Incheon National University**

- **김지범 (Ph.D):** 출신: 인천광역시 미추홀구, 서인천고등학교 졸업
- Education: 연세대학교 전기전자공학부 학사, 석사
- (미) 펜실베니아 주립대학교 컴퓨터공학과 박사 (2012)
- (미) 로스알라모스 국립연구소 Post Doc (2013)
- 현재: 인천대학교 컴퓨터공학부 정교수 (2013 – 현재)

- 연구 분야: 수치 해석, 기하 딥러닝, 인공지능 응용
- 수상: 2019, 2020 인천대 학술연구상 수상
- 과제: 현재 연구재단 중견연구, 기초연구실 과제 수행 중. 이전 연구재단 신진연구, 중기부 과제, 산업체 과제등 수행 경험
- 특허, 기술이전 실적
- 최근 5년간 기술이전 2건, 총 3000만원 기술이전 실적
- 최근 5년간 특허 등록: 10건 이상
- 인천과학예술영재학교 학생들 STEAM 과제 지도

- **오늘 실습 보조 학생**
- **박진영: 인천대 컴퓨터공학부 3학년**
- **최한슬: 인천대 컴퓨터공학부 3학년**

- **연구실 인공지능 장비**
- **Multi-GPGPU Data Processing System**
- **1. Nvidia Tesla GPU-V100 6기 (학과 공동 운용)**
- **NVIDIA V100 | NVIDIA**
- **2. Nvidia Titan XP 8기x2 (16기) 3. Nvidia A6000 GPU 3기**
- **총 1억원 이상의 장비. 7호관 서버실에 위치함 (원하는 학생들은 내일 직접 볼 수 있음)**

# 특강 내용

**1월 11일 (수) 특강**

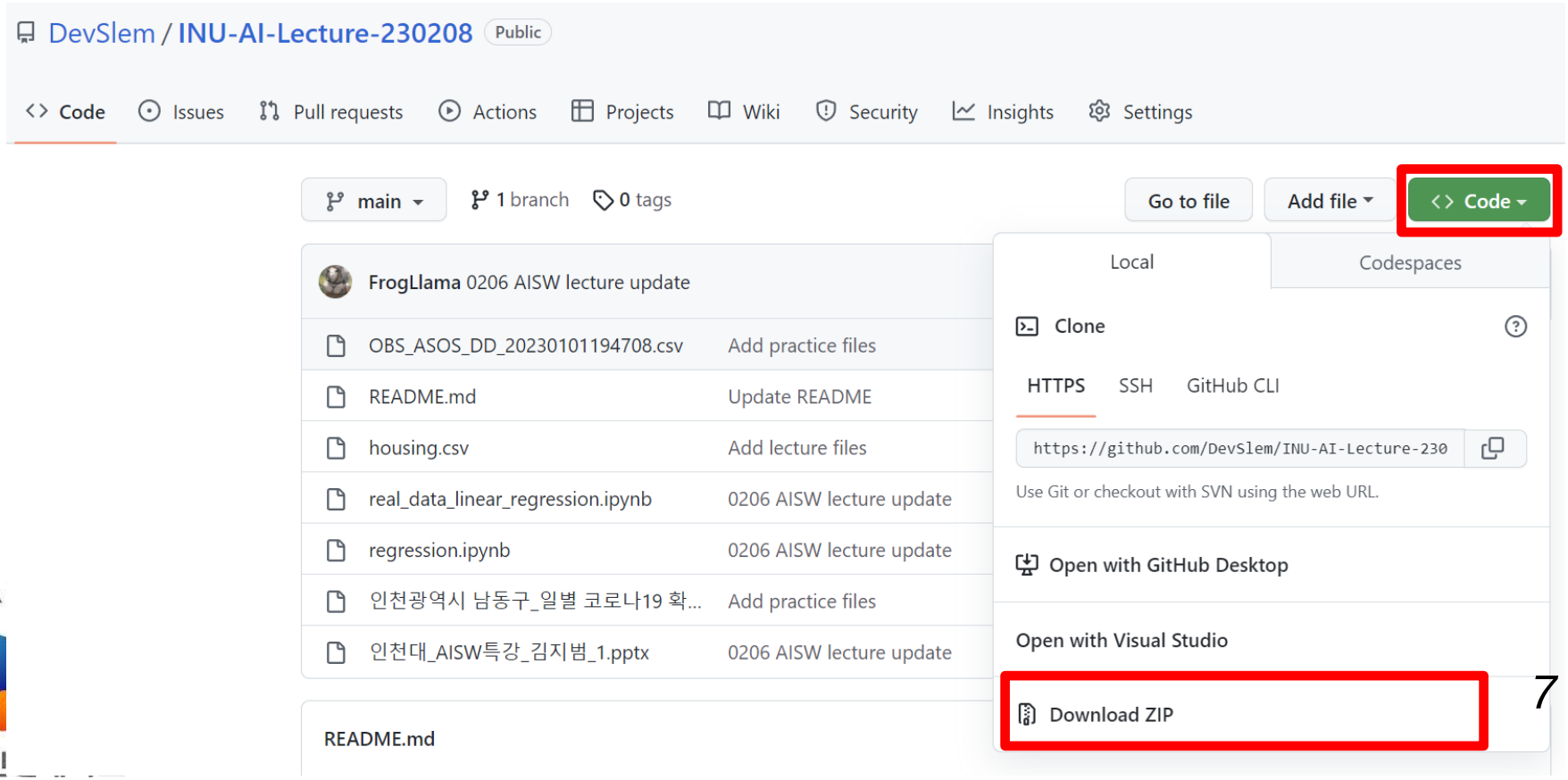
**오전**

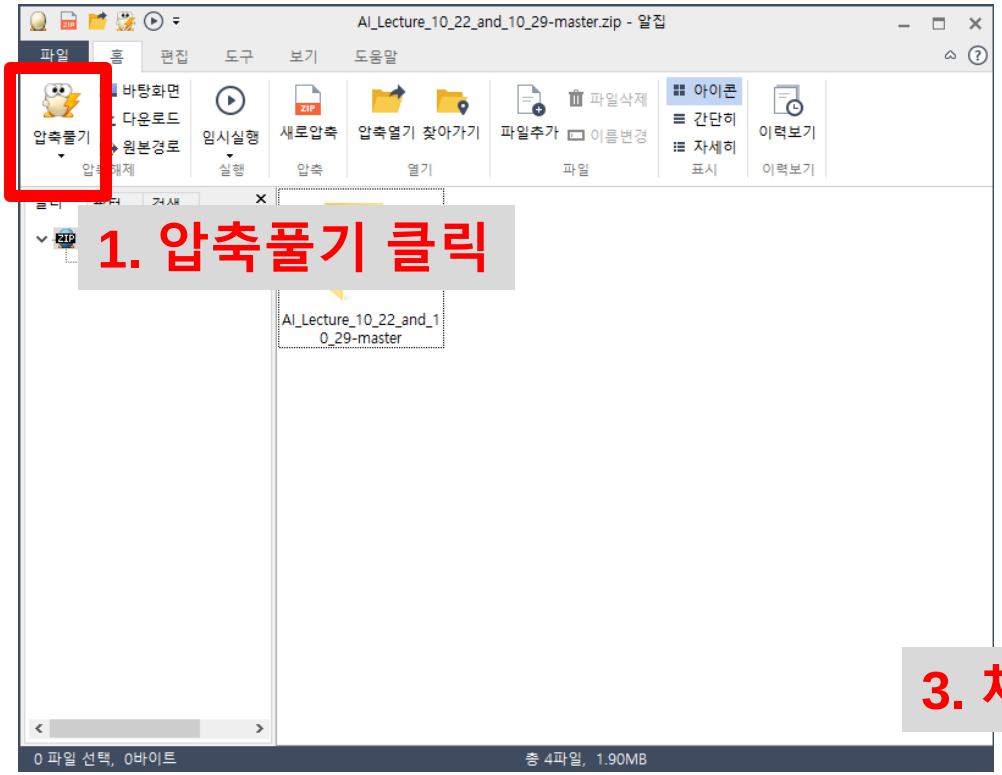
1. Machine learning과 Python 라이브러리 소개
2. Jupyter Notebook 사용법 익히기
3. 기초적인 변수 개념 및 Python 맛보기
4. MNIST dataset 맛보기 및 간단한 이진 분류기 실습

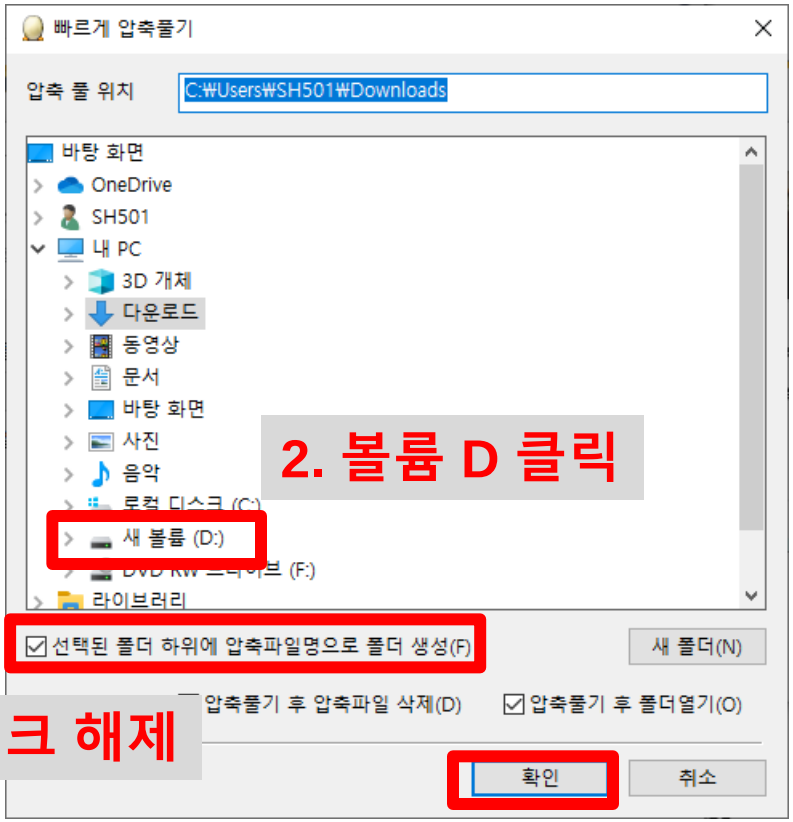
**오후**

1. Linear regression 개념 및 실습
2. Real data를 활용한 linear regression 실습

- https://url.kr/ecotl8 url에 접속
- 아래 초록색 버튼 클릭 후 Download ZIP 버튼 클릭해서 파일 다운로드






3. 체크 해제

4. 확인 클릭



인천대학교

- **1. Machine learning과 MNIST dataset**

- **Machine learning (ML)의 정의**
- **Machine learning (기계 학습) is the science (and art) of programming computers so they can learn from data**
- **즉, 주어진 데이터로부터 학습을 하도록 컴퓨터를 프로그램 하는 것**
- **또는, 컴퓨터가 주어진 데이터로부터 학습을 하는 능력을 갖도록 하는 것**

- 대표적인 ML 작업
- 분류기 (classification, predicting class)
- 회귀분석 (regression, predicting values)
- 예: MNIST dataset을 이용한 분류기

- **관련 Python library 소개**

- **NumPy**
- **Pandas**
- **Matplotlib**
- **Scikit-learn**

- **NumPy: Python에서 벡터, 행렬 등 수치 연산을 수행하는 선형대수 라이브러리**
- **사용:**

```
import numpy as np
```

- **1차원 배열: a=np.array([0.5, 1.5, 2.5])**
- **2차원 배열 (행렬): b=np.array([[0,1,2],[3,4,5]])**
- **출력**
- **print(a)**
- **print(b)**

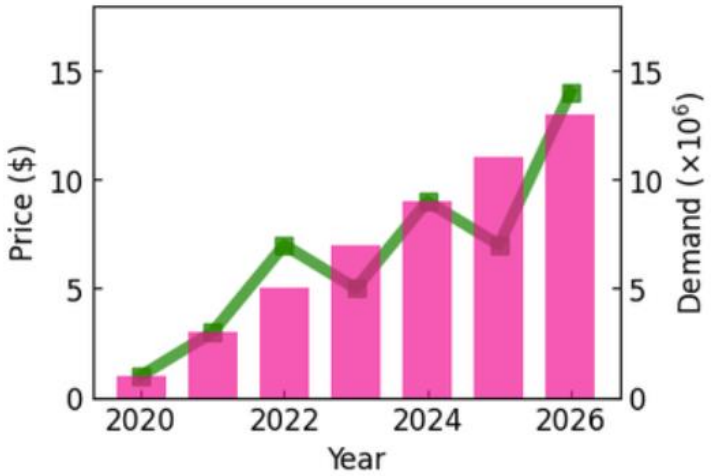
- **Pandas: 데이터 처리와 분석을 위한 라이브러리**
- **행과 열로 이루어진 데이터를 만들어 다룰수 있음**
- **Pandas 자료구조**
- **1차원: series, 2차원: DataFrame**
- **사용:**

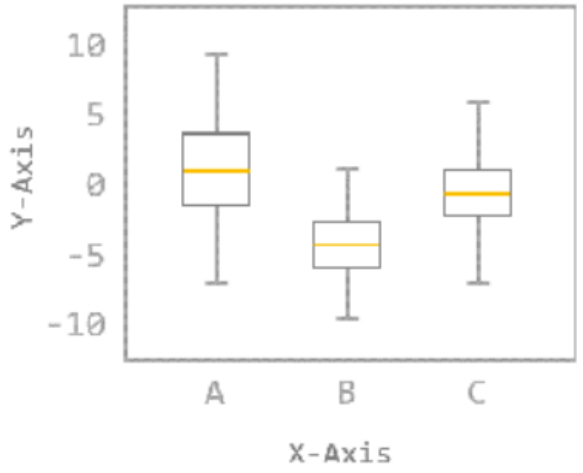
```
import pandas as pd
```

- **후에 실제 데이터를 입력 받아 데이터 분석시에 사용 예정**
- **예: csv파일**

- **Matplotlib: 대표적인 Python의 데이터 시각화 라이브러리**

```
import matplotlib.pyplot as plt
```



- **Scikit-learn: Python 기반의 machine learning 분석에 가장 많이 사용되는 툴**
- **scikit-learn: machine learning in Python — scikit-learn 1.2.0 documentation**
- **Classification, regression, clustering등 대부분의 중요한 ML 분석 가능**

- **MNIST dataset**
- 미국 고등학생과 인구조사국 직원들이 손으로 쓴 70,000개의 숫자 이미지로 구성된 데이터셋
- 사용된 0부터 9까지의 숫자는 각각 28x28= 784크기의 픽셀로 구성된 이미지 데이터
- 2차원 배열이 아닌 길이가 784인 1차원 배열로 제공
- Label: 총 70,000개의 사진 샘플이 표현하는 값
- Scikit learn 라이브러리에 MNIST 데이터셋이 있음

- **행렬 (matrix):** 수의 직사각형 **배열**. **m개의 행과 n개의 열**을 갖는 행렬을 **m × n 행렬**이라 부른다. **복수: matrices**

- 행렬 A의 i번째 행은 $[a_{i1}\ a_{i2}\ \ldots\ a_{in}]$, 행렬 A의 j번째 열은: $\begin{bmatrix} a_{1j} \\ a_{2j} \\ \ldots \\ a_{mj} \end{bmatrix}$

- 행렬 안의 값들을 **요소 (entry) 혹은 원소 (element)**라 한다

- 행렬 A의 i번째 행과 j번째 열의 요소 $(a_{ij})$는 행렬 A의 ij번째 요소라 한다

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \ldots & a_{1j} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2j} & \ldots & a_{2n} \\ \vdots & \vdots & & \vdots & & \vdots \\ a_{i1} & a_{i2} & \ldots & a_{ij} & \ldots & a_{in} \\ \vdots & \vdots & & \vdots & & \vdots \\ a_{m1} & a_{m2} & \ldots & a_{mj} & \ldots & a_{mn} \end{bmatrix} \leftarrow i\text{th row of A}$$

$$\uparrow$$

$j$th column of A

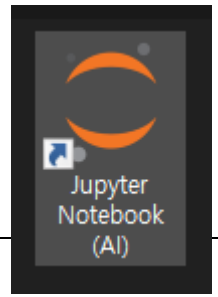
- **Jupyter Notebook 사용법**

- 오늘 실습에서는 Python 코드를 사용 예정
- Jupyter Notebook을 사용 예정

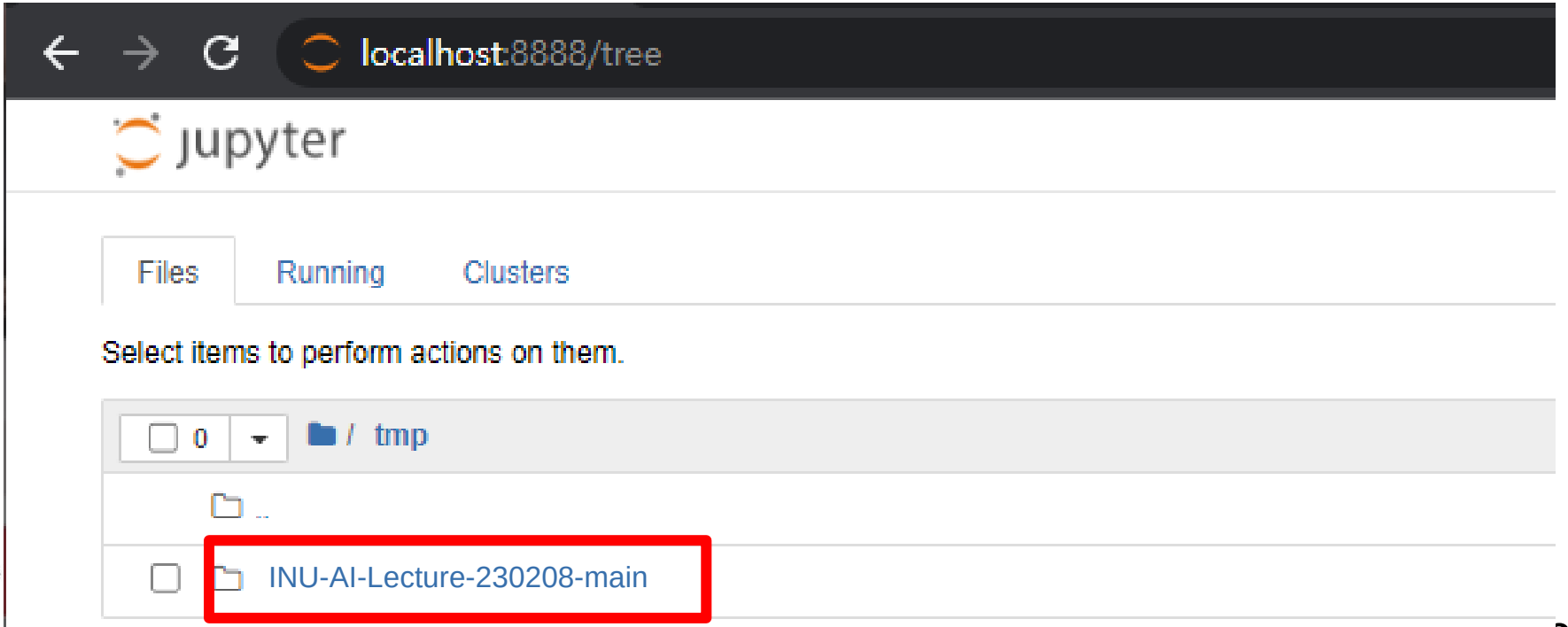
The Jupyter Notebook

The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.

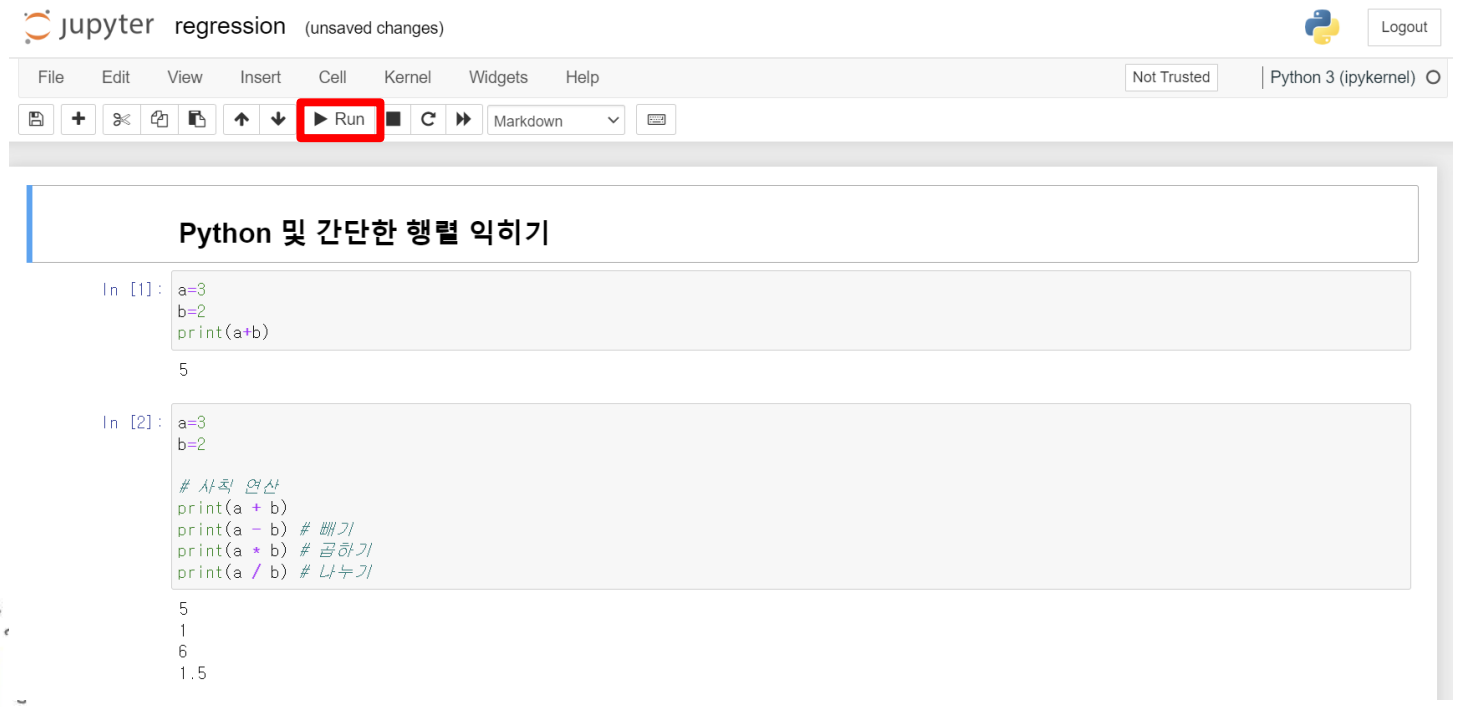
- 바탕화면의 "Jupyter Notebook (AI)" 실행



- 크롬 브라우저에 표시되는 Jupyter notebook
- 실수로 크롬 탭을 껐을 경우 크롬 주소창에 “localhost:8888” 입력
- 압축 푼 “INU-AI-Lecture-230208-main” 폴더 클릭



- “regression.ipynb” 클릭
- 실행하고 싶은 코드블럭(Cell) 클릭-> 상단 바의 “Run” 클릭



- 실행 중 (별 표시)


- 실행완료 (숫자 표시)

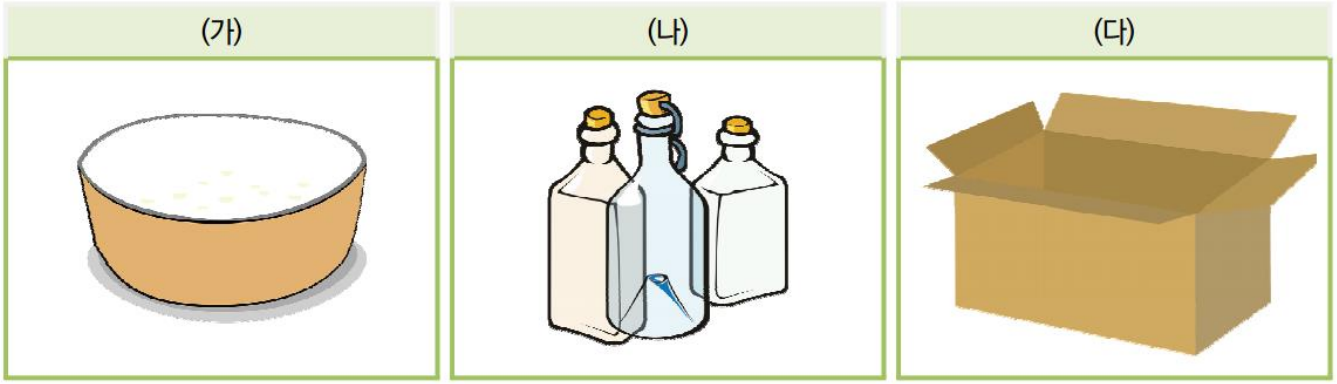
- 코드블럭(Cell)의 출력 결과는 블록 하단에 나타남
- (코드에 따라서 실행이 끝나고 출력이 없을 수도 있음)

```
    print("Saving figure", fig_id)
    if tight_layout:
        plt.tight_layout()
    plt.savefig(path, format=fig_extension, dpi=resolution)
```

```
==> WARNING: A newer version of conda exists. <==
  current version: 4.9.1
  latest version: 4.14.0

Please update conda by running

    $ conda update -n base -c defaults conda


Collecting package metadata (repodata.json): ...working... done
Solving environment: ...working... done

# All requested packages already installed.
```

- **기초적인 변수 개념 및 Python 맛보기**

# 프로그래밍: 변수 (variable)란?




*출처: pixabay.com 무료 이미지

여러분이 만약 밥을 먹으려 한다면 위의 도구 중 어떤 것을 사용할 것 같나요? 아마 (가) 가 가장 익숙할 것입니다. 만약 음료수를 담는다면 (나) 가, 책과 같은 물건을 담는다면 (다) 가 가장 적절할 것입니다. 밥, 음료수, 책과 같은 내용물을 데이터라고 한다면, 이 내용물을 담는 도구는 저장 공간이라 할 수 있습니다. 프로그래밍에서는 이렇게 데이터를 저장하는 저장 공간을 변수라고 합니다.

변수를 만들기 위해 변수의 이름과 값을 정해야 합니다. 숫자 3을 저장하는 변수 a와 숫자 2를 저장하는 변수 b를 만들어 두 변숫값의 합을 출력해봅시다.

- 실습
- 변수를 만들기 위해 변수의 이름과 값을 정해야 합니다. 숫자 3을 저장하는 변수 a와 숫자 2를 저장하는
- 변수 b를 만들어 두 변숫값의 합을 출력해봅시다
- 값을 변수에 저장, =
- 결과 출력, ‘print’ 함수 사용

| 코드 | 실행 결과 |
| --- | --- |
| a = 3<br>b = 2<br>print(a + b) | 5 |

# Python 사칙연산 및 비교연산

```
a=3
b=2

# 사칙 연산
print(a + b)
print(a - b) # 빼기
print(a * b) # 곱하기
print(a / b) # 나누기
```

```
# 비교 연산
print(a > b) # 크다
print(a < b) # 작다
print(a >= b) # 크거나 같다
print(a <= b) # 작거나 같다
print(a == b) # 같다
print(a != b) # 다르다
```

- **Python의 자료형**

- 변수에 값을 저장시에 값이 순자인가 문자열인가 등에 따라 값을 컴퓨터에 저장하는 방식이나 계산하는 방법이 다르다
- 이러한 값의 종류를 자료형 (data type)이라 한다
- 변수나 값의 자료형을 보려면 ‘type’이라고 치면 된다
- Python에서 많이 사용되는 자료형은 list, tuple, dict등이 있다
- 일단 list 자료형은 배열 생성시에 많이 사용됨

- 실습
- Python에서 list 자료형 1차원 배열 생성
- ‘len’ list의 길이 출력
- Python에서의 주석: ‘#’ 1줄 주석, 실행 안되고 설명 적음

```
1 # Python에서 많이 쓰는 list 자료형,  집합,  배열등
2 a=[1, 2, 3]
3 print(a)
4 # 변수 a의 자료형은?
5 print(type(a))
6 # 첫번째 요소 출력
7 print(a[0])
8 # list의 길이 출력
9 print(len(a))
```

```
[1, 2, 3]
<class 'list'>
1
3
```

- Python에서의 1차원 행렬 및 2차원 행렬
- Numpy library 사용, 1행 3열 및  2행 3열짜리 행렬 정의
- 주의: Python에서 기본 index는 0행, 0열 부터이다
- 행렬의 크기 출력 'shape'

```
1  # Numpy로 1차원 행렬 정의
2  import numpy as np
3  a=np.array([1,2,3])
4  print(a)
5  print(a[0])
```

```
[1 2 3]
1
```

```
1  import numpy as np
2  a=np.array([[1, 2, 3], [4, 5, 6]])
3  print(a)
```

```
[[1 2 3]
 [4 5 6]]
```

- 컴퓨터 공학에서 행렬의 사용 예
- Grayscale image (그레이 스케일 영상): 영상의 화소값을 그레이 음영으로 나타냄
- 화소값을 0부터 255사이의 부호 없는 비트 정수로 표현 (0: 흑색, 255: 흰색)

| | | | | |
|---|---|---|---|---|
| 147 | 146 | 148 | 150 | 153 |
| 145 | 149 | 151 | 154 | 156 |
| 149 | 152 | 153 | 156 | 157 |
| 150 | 153 | 155 | 157 | 158 |
| 149 | 151 | 152 | 156 | 159 |

- **Python에서 간단한 그래프 그리기**

- **Matplotlib**
- **데이터를 차트 (chart)나 플롯 (plot)으로 시각화하는 라이브러리**
- **1. 라인 플롯 (선그래프) 그리기**
- **다양한 옵션 가능**

```
1  import matplotlib.pyplot as plt
2  x = [1, 2, 3, 4]
3  y = [2, 4, 6, 8]
4  plt.plot(x, y, '-o')
```

- 많은 경우에 x축을 등간격으로 나눈후 그 x값에서의 y값을 구한 후 선 그래프를 그린다
- 이때 ‘linspace’ 함수가 흔히 사용된다
- 아래 예는 x를 [0, 10]까지 등간격으로 200개의 점을 사용하여 샘플링한 후 그때 y=sin(x)를 구한 후 라인 그래프로 그린 예이다

```
import matplotlib.pyplot as plt
import numpy as np
x=np.linspace(0, 10,200)
y=np.sin(x)
plt.plot(x, y, '-r')
```

## 2. 축 레이블 삽입하기

```
1  import matplotlib.pyplot as plt
2  import numpy as np
3  x=np.linspace(0, 10,200)
4  y=np.sin(x)
5  plt.plot(x, y, '-r')
6  plt.xlabel('x axis')
7  plt.ylabel('y axis')
```

- **간단한 Pandas 맛보기**

- **Pandas:** 데이타 분석을 위해 널리 사용되는 파이썬 라이브러리 패키지
- 1차원: ‘series’
- 2차원: ‘dataframe’
- 2차원 행렬형태에서 많이 사용되는 ‘dataframe’ 실습

- **Python의 pandas의 dataframe 활용**
- **head(): 데이터의 상위 N행 봄. 기본 N=5**
- **info(): 데이터에 대한 전반적인 정보**
- **describe(): 열별 요약 통계량 (수치형만)**
- **hist(): histogram 보기**

- **MNIST Dataset 맛보기**

- **MNIST dataset의 문제 정의**
- **지도학습: 각 이미지가 담고 있는 숫자가 레이블(label)로 지정됨**
- **분류: 이미지 데이터를 분석하여 0부터 9까지의 숫자로 분류**
- **이미지 그림을 총 10개의 클래스로 (class)로 분류하는 multiclass classification**

- Training data와 test data 나누기
- MNIST dataset은 이미 6:1로 분류되어 있음
- 훈련세트: 앞쪽 60,000개 이미지
- 테스트세트: 나머지 10,000개 이미지

- **Binary classification**
- **이진 분류기: “5” 인가? “5”가 아닌가?**

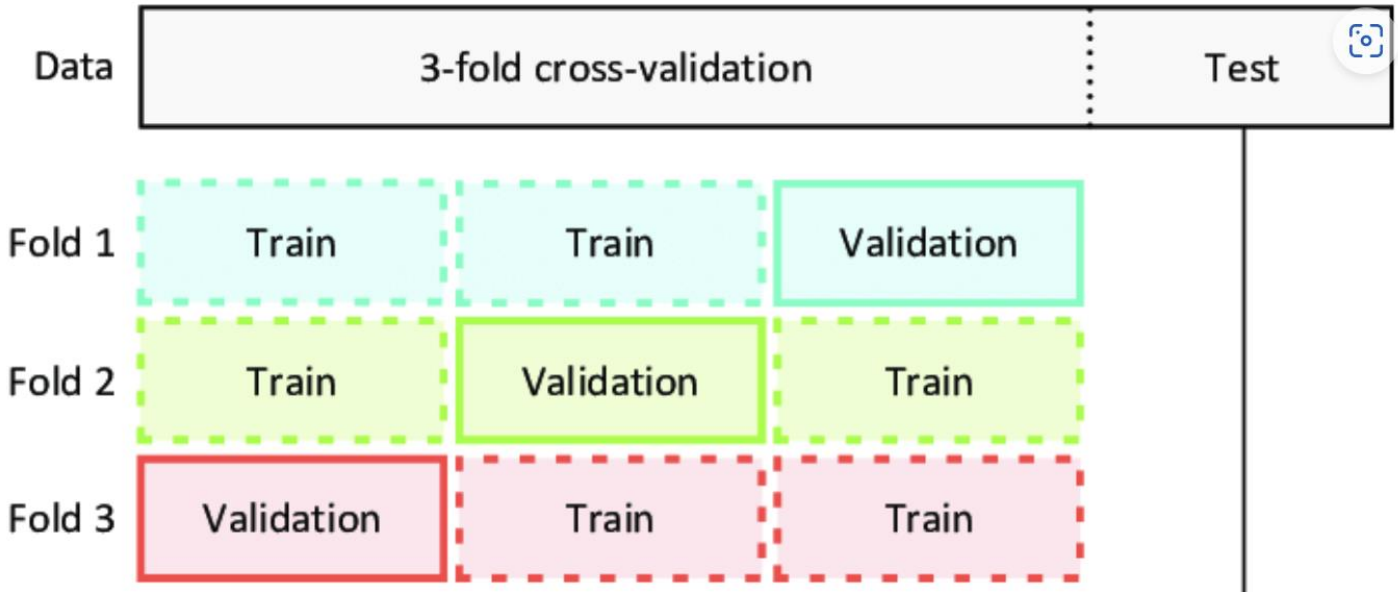
In [14]:
```
y_train_5 = (y_train == 5)
y_test_5 = (y_test == 5)
```

- **분류기: stochastic gradient descent (SGD) classifier 사용 (Scikit-learn 제공)**

In [15]:
```
from sklearn.linear_model import SGDClassifier

sgd_clf = SGDClassifier(max_iter=1000, tol=1e-3, random_state=42)
sgd_clf.fit(X_train, y_train_5)
```

- 분류기 성능 평가: K-fold Cross validation (CV)
- 왜 필요한가? 학습한 dataset에 평가까지 하면 왜곡된 결과가 나올 수 있음 (학습하지 않은 데이터에 test를 해봐야함). Overfitting 문제
- fold: 부분집합 (subset)
- 한쪽 데이터에 치우친 결과를 방지하기 위하여
- 모든 데이터의 한 부분이 학습에 사용되고 test에도 사용되면 fair함

- **Training data를 3개 부분으로 나누고 이 3개 부분 중의 2개 부분을 학습용으로 쓰고 나머지 1개 부분을 검증 (validation) 용으로 사용**
- **훈련 (학습) 데이터의 모든 부분이 학습과 검증에 모두 사용**
- **결과가 타당하다면 이 모델을 최종 test에 사용**



- **Above 95% accuracy on all cross-validation folds (이경우 3-fold CV 사용)**

```
In [17]: from sklearn.model_selection import cross_val_score
         cross_val_score(sgd_clf, X_train, y_train_5, cv=3, scoring="accuracy")

Out[17]: array([0.95035, 0.96035, 0.9604 ])
```

- **Good? Let's look at a very dumb classifier that just classifies every single image in the "not-5" class**

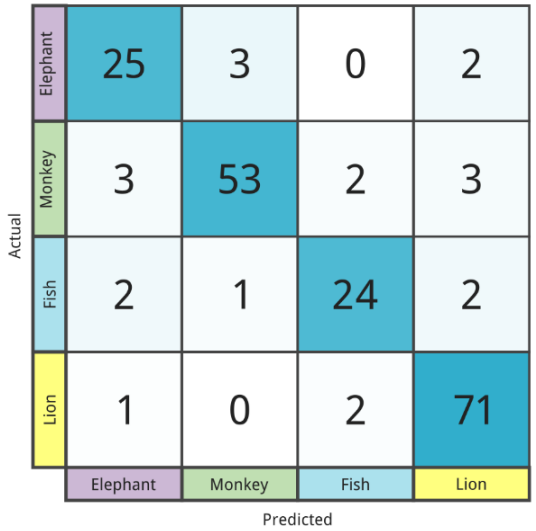
- **It has over 90% accuracy? Why?**
- **Because only about 10% of the images are 5s, so if you always guess that an image is not a 5, you will be right about 90% of the time**

In [19]:
```
from sklearn.base import BaseEstimator
class Never5Classifier(BaseEstimator):
    def fit(self, X, y=None):
        pass
    def predict(self, X):
        return np.zeros((len(X), 1), dtype=bool)
```

In [20]:
```
never_5_clf = Never5Classifier()
cross_val_score(never_5_clf, X_train, y_train_5, cv=3, scoring="accuracy")
```

Out[20]:
```
array([0.91125, 0.90855, 0.90915])
```

- This demonstrates why accuracy is generally not the preferred performance measure for classifiers
- 오차 행렬 (confusion matrix)
- 클래스별 예측 결과를 정리한 행렬
- 오차행렬의 행은 실제 class를 열은 예측된 class를 가리킴



- **The first row of this matrix considers non-5 images: 53,892 of them were correctly classified as non-5s, while 687 were wrongly classified as 5s**
- **1,891 were wrongly classified as  non-5s, 3,530 were correctly classified as 5s**

In [21]:
```
from sklearn.model_selection import cross_val_predict

y_train_pred = cross_val_predict(sgd_clf, X_train, y_train_5, cv=3)
```

In [22]:
```
from sklearn.metrics import confusion_matrix

confusion_matrix(y_train_5, y_train_pred)
```

Out[22]:
```
array([[53892,   687],
       [ 1891,  3530]])
```

- 완벽한 분류기
- 오차 행렬의 대각부분 (diagonal)에만 값이 있음

In [23]:
```
y_train_perfect_predictions = y_train_5  # pretend we reached perfection
confusion_matrix(y_train_5, y_train_perfect_predictions)
```

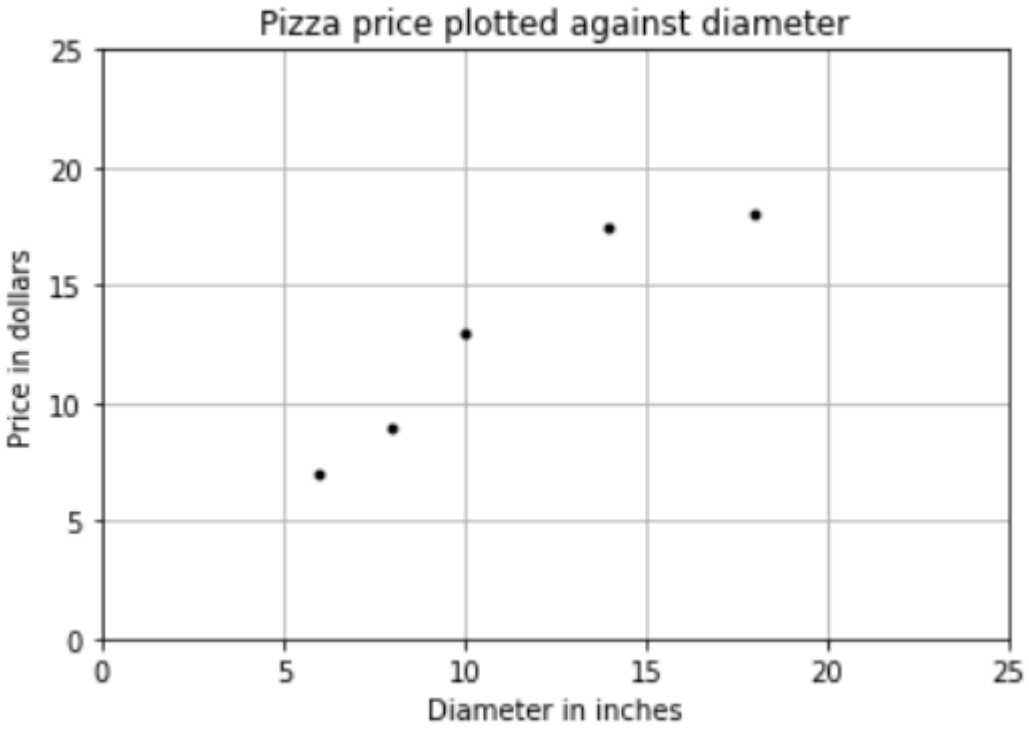
Out[23]:
```
array([[54579,     0],
       [    0,  5421]])
```

- **Linear regression (선형 회귀분석)**

- **Problem:** 만일 당신이 아래와 같이 5개의 training instance (sample)가 주어져 있을 때, 즉, 5번 피자를 먹었을 때 피자의 크기별로 가격이 주어져 있다.
- 목표: 여기 주어져 있지 않은 피자 크기에 대해서, 즉, 일반적으로 피자의 크기에 대해서 가격이 얼마정도 되는지 예측해보고자 한다

| Training sample | 크기 (inches) | 가격 (dollar) |
|---|---|---|
| 1 | 6 | 7 |
| 2 | 8 | 9 |
| 3 | 10 | 13 |
| 4 | 14 | 17.5 |
| 5 | 18 | 18 |

- 실습 (그래프)



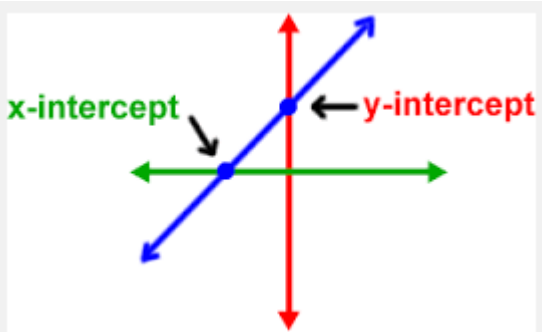
- 앞의 그래프를 보면 피자의 크기와 가격이 양의 상관관계에 있다는 것을 알수있다.
- 즉, 피자의 크기가 커지면, 가격도 같이 증가한다는 것이다
- Linear regression은 기본적으로
- $y = \alpha + \beta x$
- 입력: $x$ (여기서는 피자 크기), 출력: $y$ (여기서는 가격)
- 이 주어져 있을때 주어진 데이터를 가장 잘 근사화하도록

$\alpha, \beta$를 학습을 통해서 알아낸다

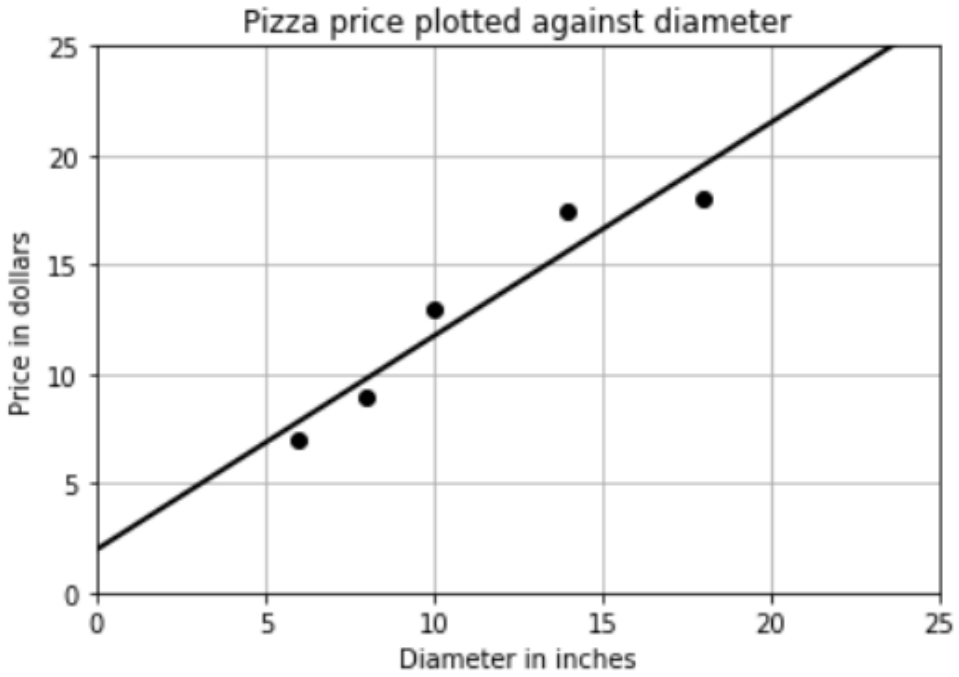
- Scikit-learn에서의 linear regression

- from sklearn.linear_model import LinearRegression
- model=LinearRegression()
- model.fit(X,y)

- ----------
- model은 두 값을 가지는데 근사화한 직선을 y=ax+b라 하면
- model.coef_ 는 그 직선의 기울기 (a)
- model.intercept_ 는 그 직선의 y절편 값(b)
- 을 출력해준다

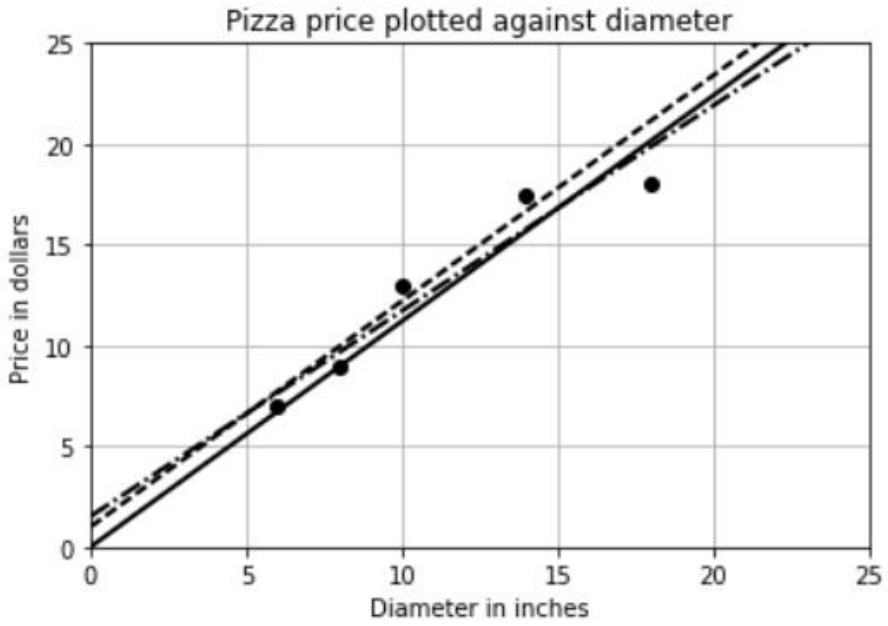


# 실습



- **Linear regression의 평가와 예측**

- 앞에서 주어진 training data를 사용하여 linear regression을 수행하였다
- How can we assess which parameters produced the best-fitting regression line?



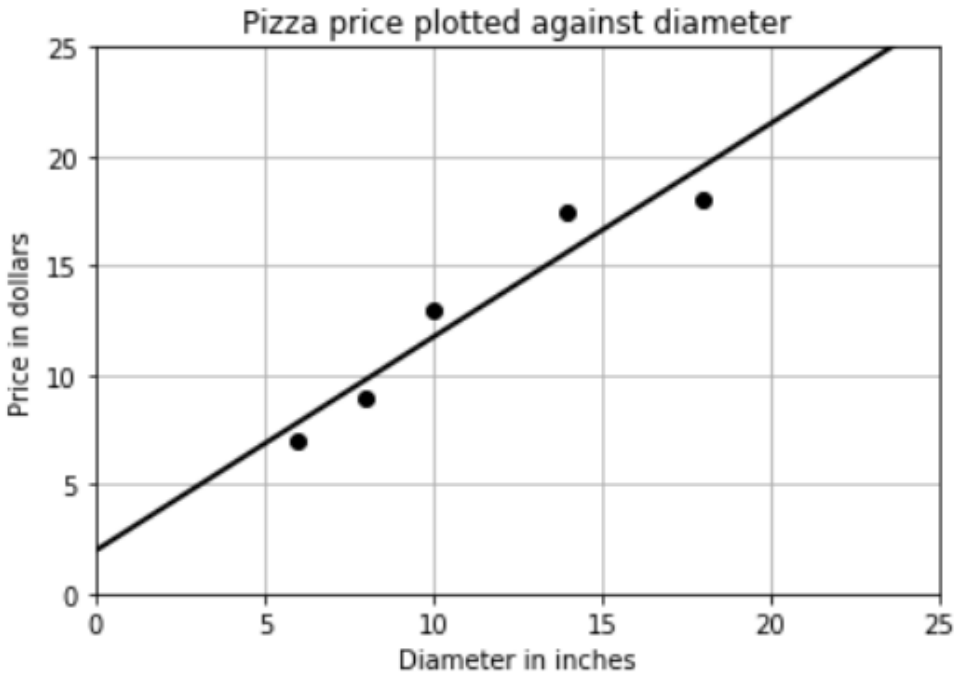
- **성능 측정 (performance measure)**
- **A typical performance measure for regression problems is the Root Mean Square Error (RMSE)**
- **RMSE 값이 크다면 오차가 큰 것임**

- $RMSE = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(\hat{y}_i - y_i)^2}$
- $y_i$: i번째 데이터의 실제 값
- $\hat{y}_i$: i번째 데이터의 예측 값
- $m$: 데이터 갯수

- 루트를 안씌운것을 MSE라고 한다
- Q: RMSE에서 왜 루트를 씌웠을까?
- RMSE 실습

- **RMSE 실습**

- Training data에 없는 값 예측 **predict()**
- **크기 12일 때 피자 가격 예측**
- **실습**




```
A 12" pizza should cost: $13.68
```

- **Multiple linear regression**

- 지금까지는 training data의 입력 특징이 1개 (피자 크기)이고 출력이 1개 (가격)인 문제를 살펴보았다
- 하지만, 피자의 가격에 영향을 미치는 것이 크기만은 아닐것이다. 예를 들어 피자에 올라가있는 토핑수가 가격에 영향을 줄 수 있다

$y = \alpha + \beta_1 x_1 + \beta_2 x_2$ (여기서 $x_1$: 크기, $x_2$: 토핑수, $y$: 가격)

- 직경, 토핑수에 따른 피자 가격의 표

| 직경 (인치) | 토핑수 | 가격 (달러) |
|---|---|---|
| 6 | 2 | 7 |
| 8 | 1 | 9 |
| 10 | 0 | 13 |
| 14 | 2 | 17.5 |
| 18 | 0 | 18 |

- 직경=8인치, 토핑수=2일때의 가격을 예측해 보자
- $\boldsymbol{x_1}$: 크기, $\boldsymbol{x_2}$: 토핑수, $\boldsymbol{y}$: 가격 이라고 하면
- 목표: 가장 잘 근사화 하는 $\boldsymbol{y = \alpha + \beta_1 x_1 + \beta_2 x_2}$ 를 찾음 (예: RMSE 기준)

- 분석해보면 가격에 피자 크기와 토핑수가 모두 양의 상관관계에 있음을 알 수 있다
- 즉, 피자 크기와 토핑수가 증가하면 가격도 올라간다
- 또한, 둘 중에 피자 크기가 가격에 영향을 더 많이 미친다는 것도 알수 있다

- **실습**

- 실제 real data를 활용한 linear regression
- 출처: start! 인공지능 with 파이썬

# 공개 데이터

| 공공 데이터 | 민간 데이터 |
| --- | --- |
| ✔ 공공데이터 포털 https://www.data.go.kr/ | ✔ 캐글 www.kaggle.com |
| ✔ 서울 열린데이터 광장 https://data.seoul.go.kr/ | ✔ Google 트렌드 https://trends.google.com/ |
| ✔ 경기데이터드림 https://data.gg.go.kr/ | ✔ Amazone 알렉사닷컴 https://www.alexa.com/ |
| ✔ 데이터 스토어 https://www.datastore.or.kr/ | ✔ 네이버 데이터랩 https://datalab.naver.com/ |
| ✔ 마이크로데이터 통합서비스 https://mdis.kostat.go.kr/index.do | ✔ SK 빅데이터 허브 https://www.bigdatahub.co.kr/ |

- 데이터가 제공되는 형태는 CSV, JSON, XLSX 등 파일 다운로드 형태와 API로 제공되는 형태로 구분할 수 있습니다.

- 이번 실습에서는 실제 데이터를 활용
- 인천시 남동구 확진자 수 데이터 (.csv)
- 데이터 상세 | 공공데이터포털 (data.go.kr)
- 인천시 기상 데이터를 활용 (.csv)
- 기상자료개방포털[데이터:기상관측:지상:종관기상관측(ASOS):자료] (kma.go.kr)

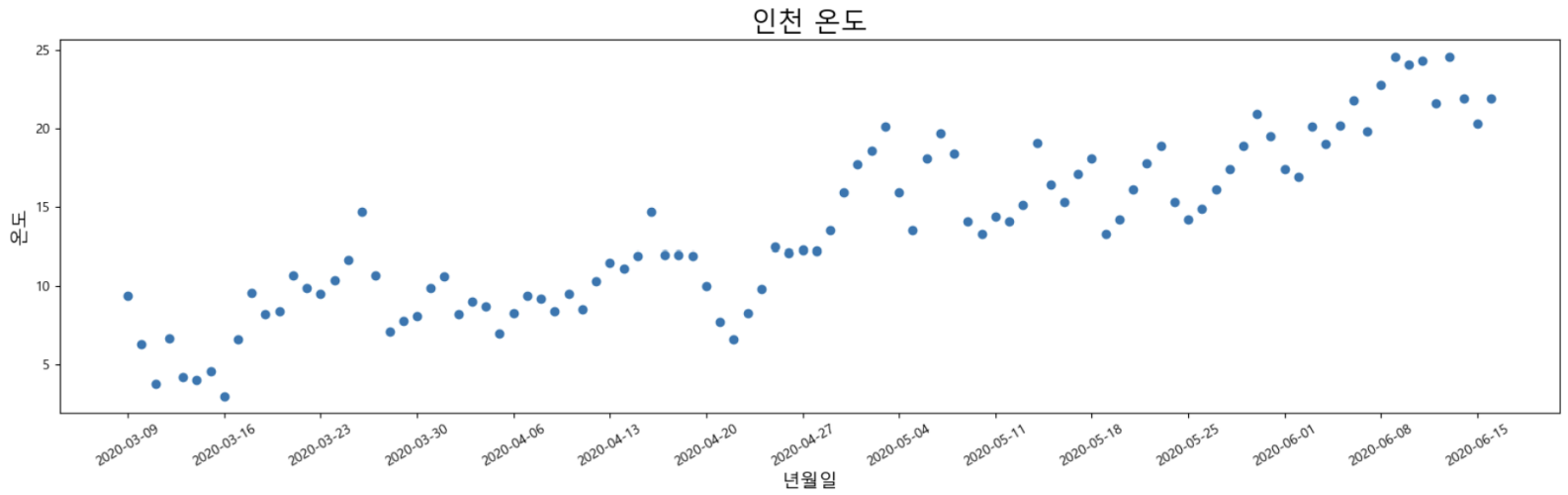
# 인천시 남동구 코로나 확진자수 데이터

```
날짜,확진자수,누적확진자수,데이터기준일
2020-03-09,1,1,2022-09-28
2020-03-10,0,1,2022-09-28
2020-03-11,0,1,2022-09-28
2020-03-12,1,2,2022-09-28
2020-03-13,0,2,2022-09-28
2020-03-14,1,3,2022-09-28
2020-03-15,1,4,2022-09-28
2020-03-16,0,4,2022-09-28
2020-03-17,0,4,2022-09-28
2020-03-18,0,4,2022-09-28
2020-03-19,0,4,2022-09-28
2020-03-20,0,4,2022-09-28
2020-03-21,0,4,2022-09-28
2020-03-22,0,4,2022-09-28
2020-03-23,0,4,2022-09-28
2020-03-24,0,4,2022-09-28
2020-03-25,0,4,2022-09-28
2020-03-26,1,5,2022-09-28
2020-03-27,0,5,2022-09-28
2020-03-28,1,6,2022-09-28
2020-03-29,0,6,2022-09-28
2020-03-30,2,8,2022-09-28
2020-03-31,1,9,2022-09-28
2020-04-01,0,9,2022-09-28
2020-04-02,0,9,2022-09-28
2020-04-03,0,9,2022-09-28
2020-04-04,0,9,2022-09-28
2020-04-05,0,9,2022-09-28
2020-04-06,0,9,2022-09-28
2020-04-07,0,9,2022-09-28
2020-04-08,0,9,2022-09-28
```

# 기상 데이터 전처리


| 년월일시 | 평균기온(°C) | 일강수량(mm) | 최대 풍속(m/s) | 평균 상대습도(%) | 일 최심적설(cm) |
|---|---|---|---|---|---|
| 2020-01-24 | 2.8 | NaN | 3.7 | 63.0 | NaN |
| 2020-01-25 | 4.3 | NaN | 3.9 | 63.1 | NaN |
| 2020-01-26 | 5.5 | NaN | 6.1 | 56.3 | NaN |
| 2020-01-27 | 5.9 | 0.1 | 7.0 | 48.1 | NaN |
| 2020-01-28 | 6.8 | 0.0 | 5.4 | 49.4 | NaN |

- 기상 데이터 histogram
- Python의 Matplotlib 라이브러리 사용
- Scatter plot (산점도)
- plt.scatter

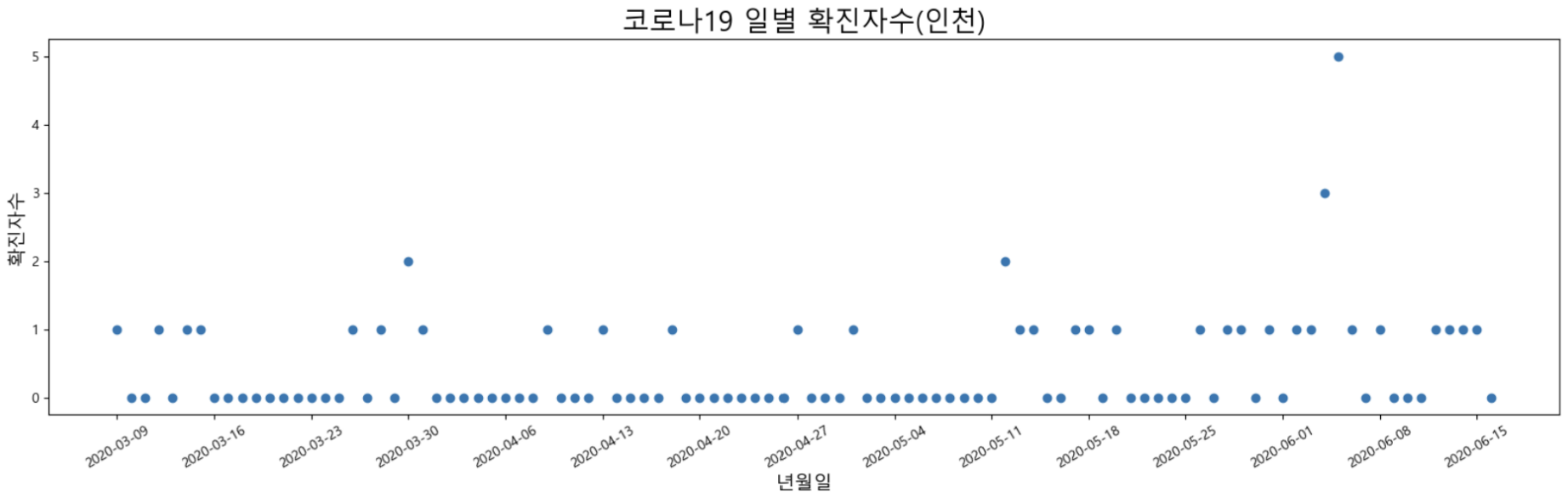





인천대학교

# 코로나 확진자수 데이터 전처리

# 코로나19 일별 확진자수 시각화






인천대학교

데이터 병합하기 → CSV 가져오기

- 두 데이터 병합하기

| 년월일시 | 평균기온(° C) | 일강수량(mm) | 최대 풍속(m/s) | 평균 상대습도(%) | 일 최심적설(cm) |
|---|---|---|---|---|---|
| 2020-01-24 | 2.8 | NaN | 3.7 | 63.0 | NaN |
| 2020-01-25 | 4.3 | NaN | 3.9 | 63.1 | NaN |
| 2020-01-26 | 5.5 | NaN | 6.1 | 56.3 | NaN |
| 2020-01-27 | 5.9 | 0.1 | 7.0 | 48.1 | NaN |
| 2020-01-28 | 6.8 | 0.0 | 5.4 | 49.4 | NaN |

병

| 년월일 | 확진자수 |
|---|---|
| 2020-01-24 | 1 |
| 2020-01-30 | 3 |
| 2020-01-31 | 3 |
| 2020-02-02 | 1 |
| 2020-02-05 | 2 |

- 데이터 전처리가 완료된 새로운 데이터

| 년월일 | 평균기온(° C) | 일강수량(mm) | 최대 풍속(m/s) | 평균 상대습도(%) | 일 최심적설(cm) | 확진자수 |
|---|---|---|---|---|---|---|
| 2020-01-24 | 2.8 | 0.0 | 3.7 | 63.0 | 0.0 | 1 |
| 2020-01-25 | 4.3 | 0.0 | 3.9 | 63.1 | 0.0 | 0 |
| 2020-01-26 | 5.5 | 0.0 | 6.1 | 56.3 | 0.0 | 0 |
| 2020-01-27 | 5.9 | 0.1 | 7.0 | 48.1 | 0.0 | 0 |
| 2020-01-28 | 6.8 | 0.0 | 5.4 | 49.4 | 0.0 | 0 |

- 최종 목표: linear regression을 사용한 날씨에 따른 코로나 19 확진자수 예측 모델 만들기
- 입력 변수: 기온, 강수, 풍속, 습도, 적설량
- 출력 변수: 코로나 19 확진자수

- 실습

- **Python의 pandas의 dataframe 활용**
- **head(): 데이터의 상위 N행 봄. 기본 N=5**
- **info(): 데이터에 대한 전반적인 정보**
- **describe(): 열별 요약 통계량 (수치형만)**
- **hist(): histogram 보기**

- **Linear regression을 이용한 미래 인구 예측**

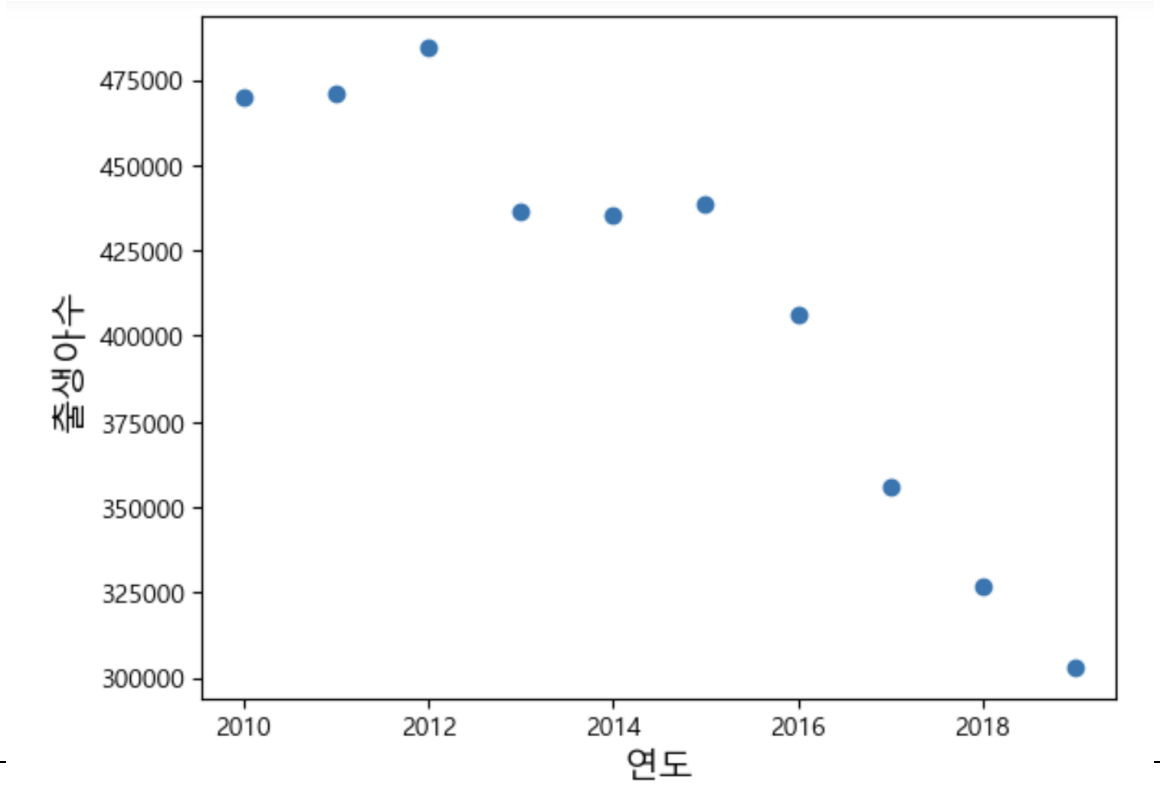
다음 2010년부터 2019년까지 우리나라 출생아수를 나타낸 표를 보고 각 물음에 답해봅시다.

연도별 출생 추이[49)]

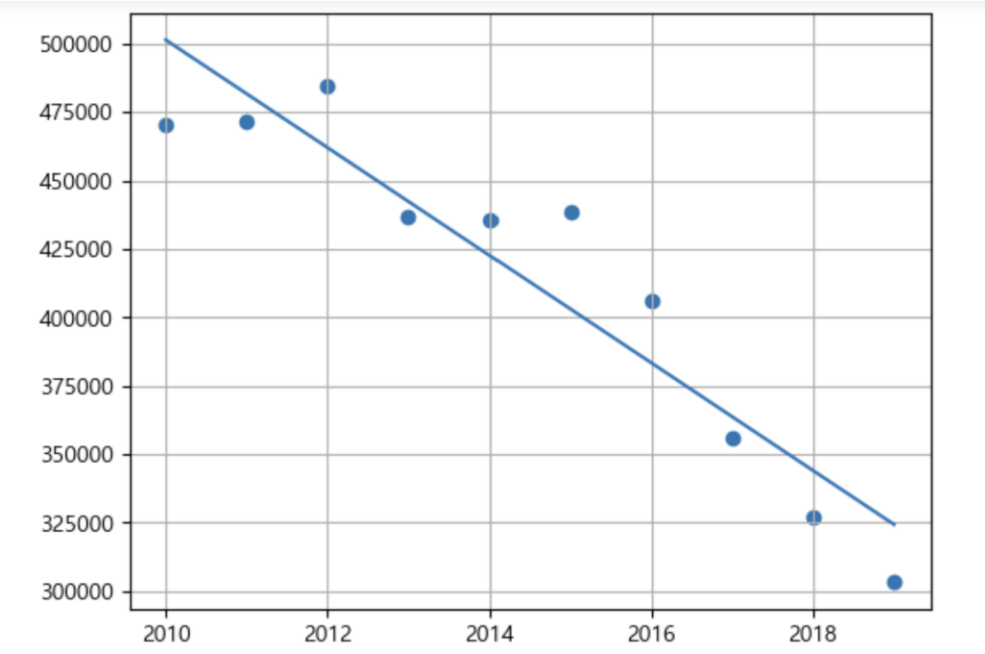
[단위 : 명, 1천명당 명, 년]

| | 2010 | 2011 | 2012 | 2013 | 2014 | 2015 | 2016 | 2017 | 2018 | 2019 |
|---|---|---|---|---|---|---|---|---|---|---|
| 출생아수 | 470,171 | 471,265 | 484,550 | 436,455 | 435,435 | 438,420 | 406,243 | 357,771 | 326,822 | 303,054 |

- **1. scatter plot을 사용하여 아래와 같이 시각화**



- **2. Linear regression을 사용하여 가장 잘 근사화 하는 직선을 찾아보고 이를 시각화 해보자**



- **3. 이때의 RMSE를 구해보자**

20916.564939838252

- **4. 2020년의 인구를 예측해보자**

304595.2666666657