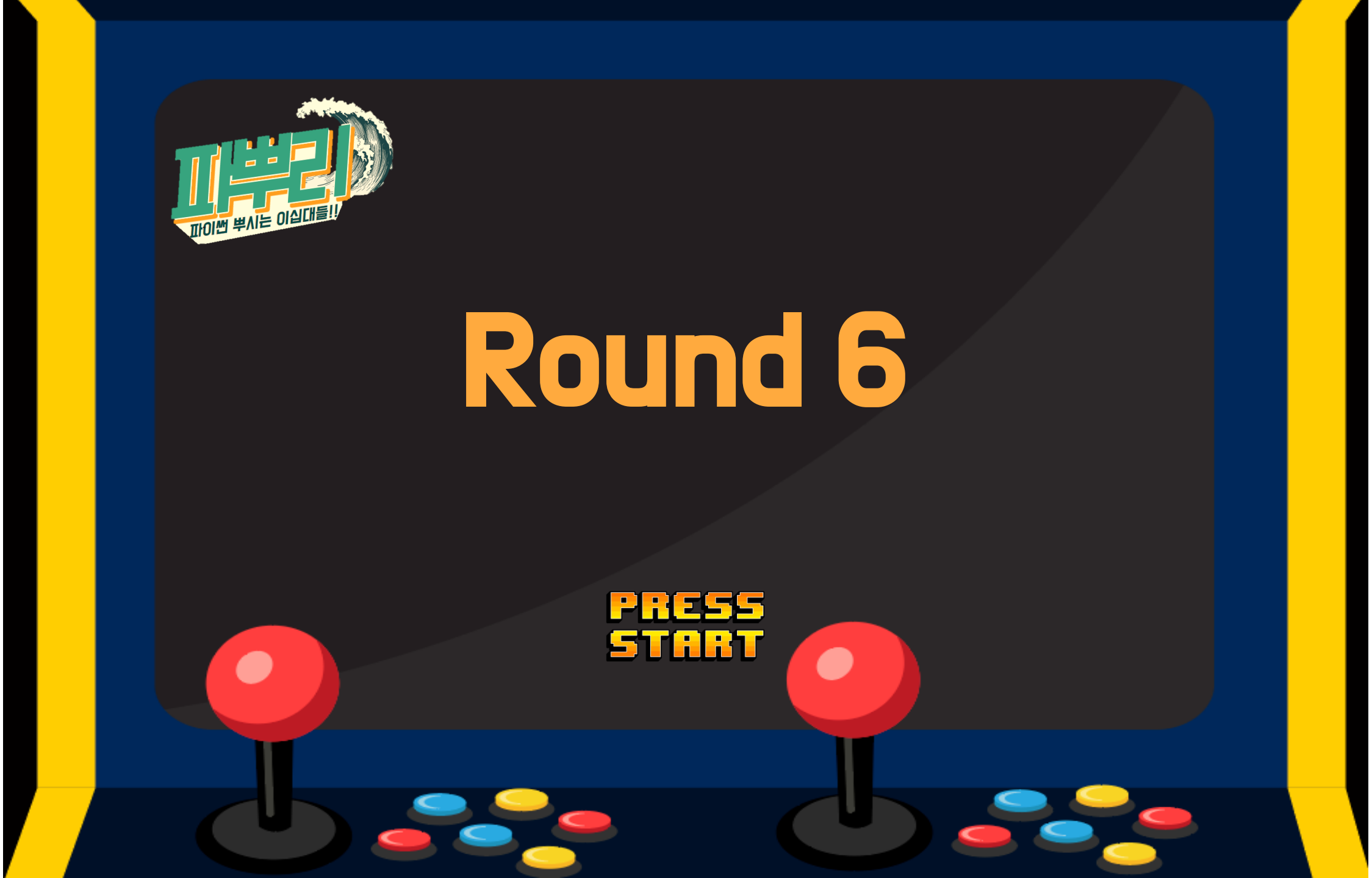

파뿌리
파이썬 뿌시는 이십대들!!
Round 6
PRESS
START

파뿌리
파이썬 뿌시는 이십대들!!

《 Round 6 》

- 머신러닝 개요
- 단순회귀분석 실습
- 다항회귀분석 실습

New Assignment

파뿌리
파이썬 뿌시는 이십대들!!

《 Round 6 》

- 머신러닝 개요 《
- 단순회귀분석 실습
- 다항회귀분석 실습

Machine Learning?

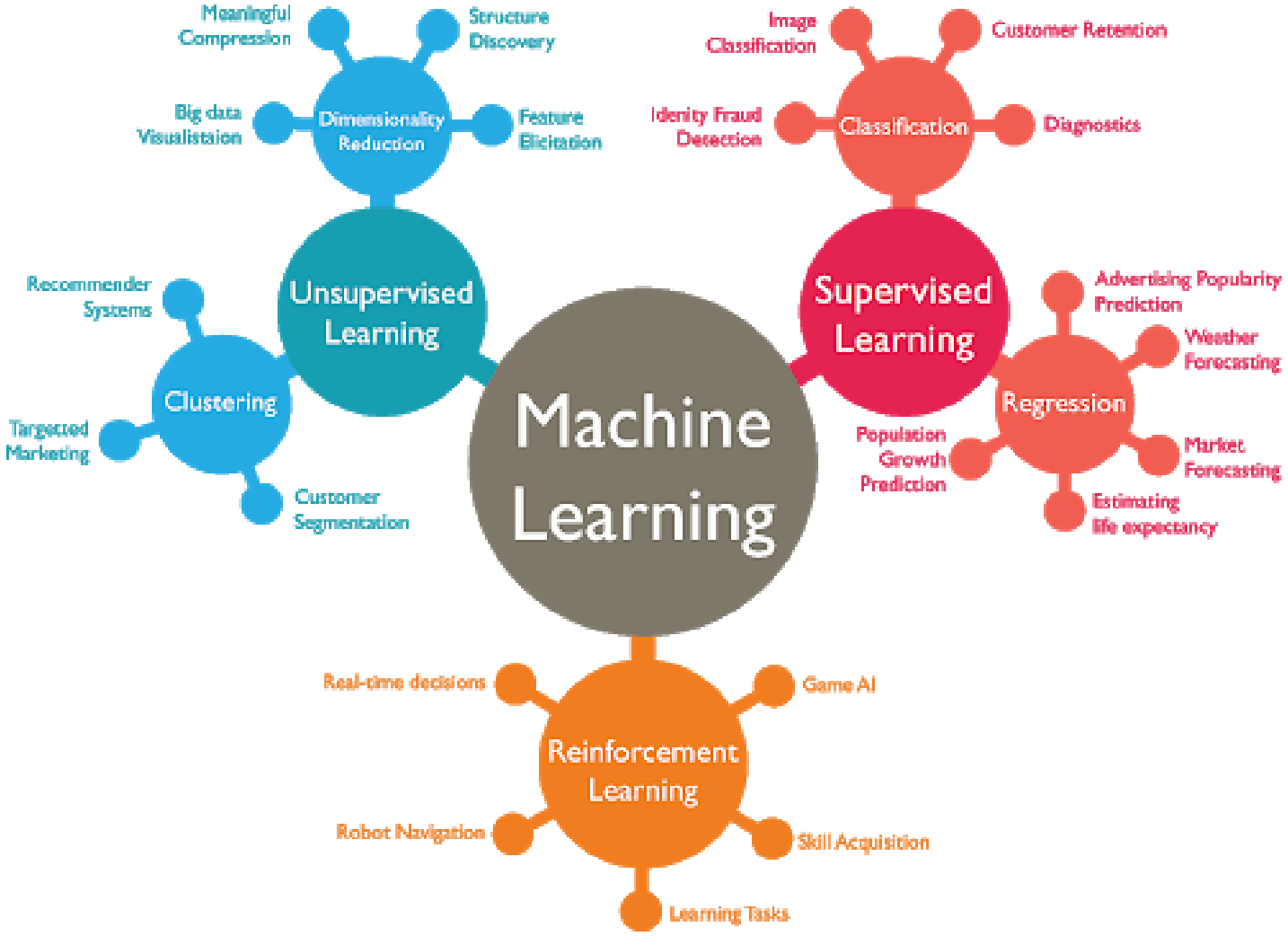
- 인공지능의 한 분야

- 어떠한 작업 T에 대해 꾸준한 경험 E를 통하여
  그 T에 대한 성능 P를 높이는 것

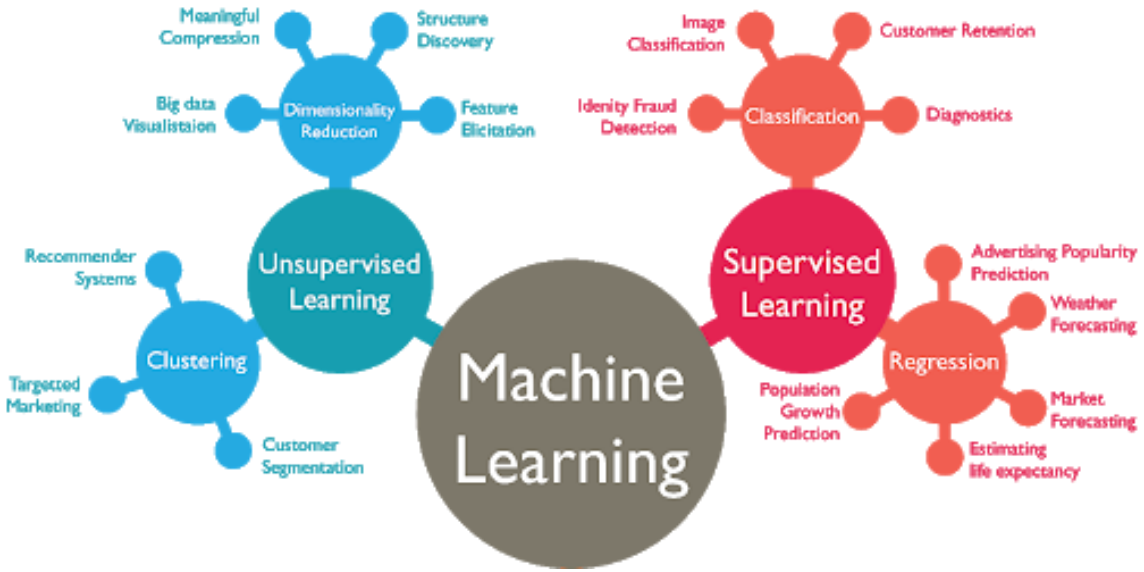
ex)

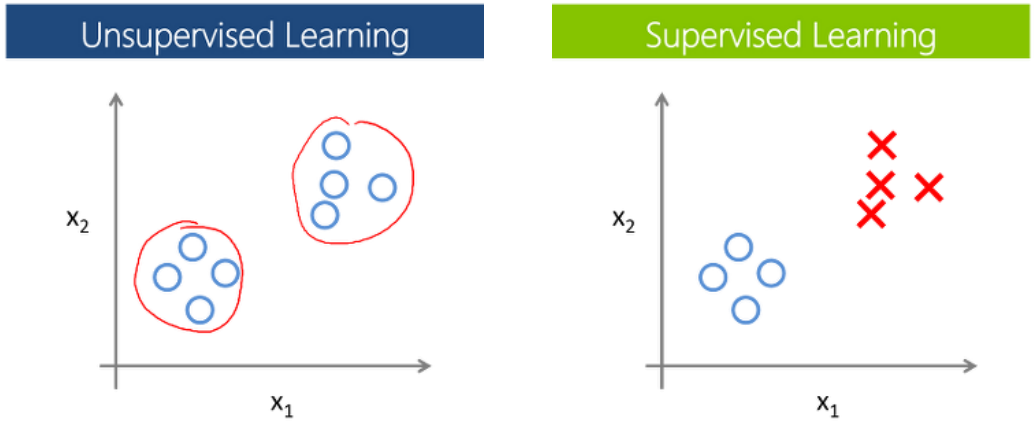
- 주가, 환율 등 경제지표 예측 (Prediction)
- 은행에서 고객을 분류하여 대출을 승인하거나 거절 (Classification)
- 비슷한 소비패턴을 지닌 고객 유형을 군집으로 묶음 (Clustering)

Kinds of Machine Learning



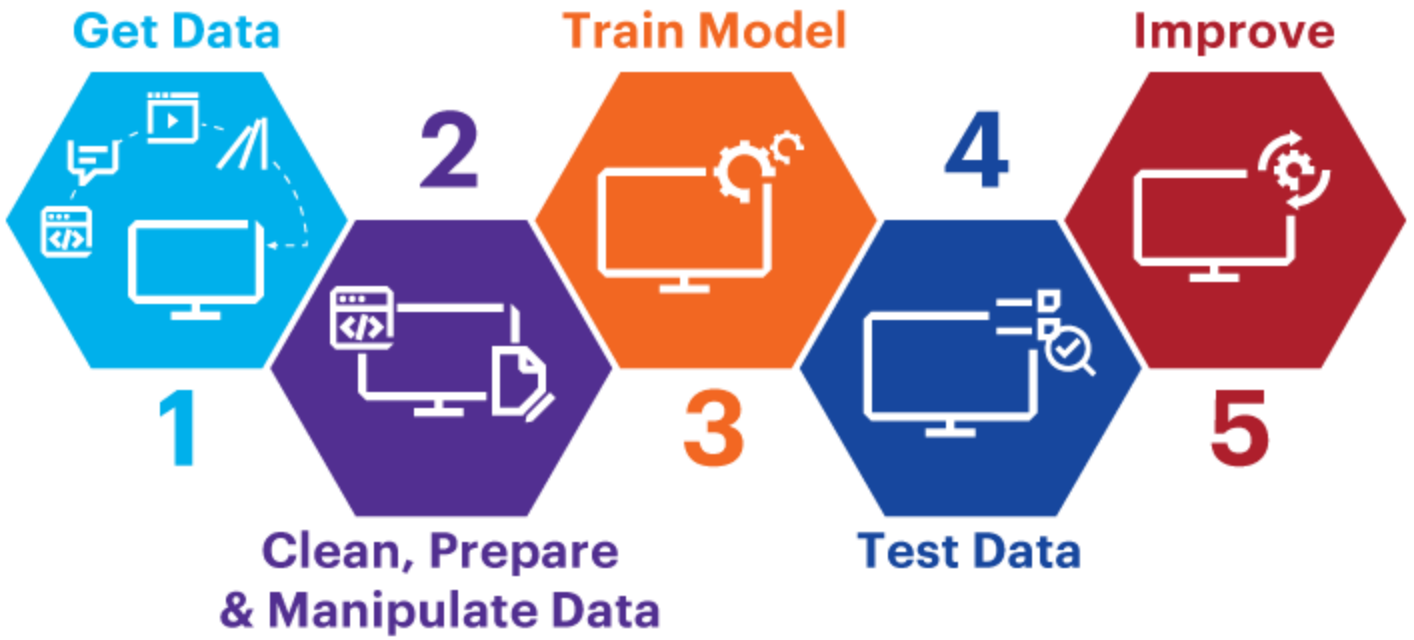
Kinds of Machine Learning



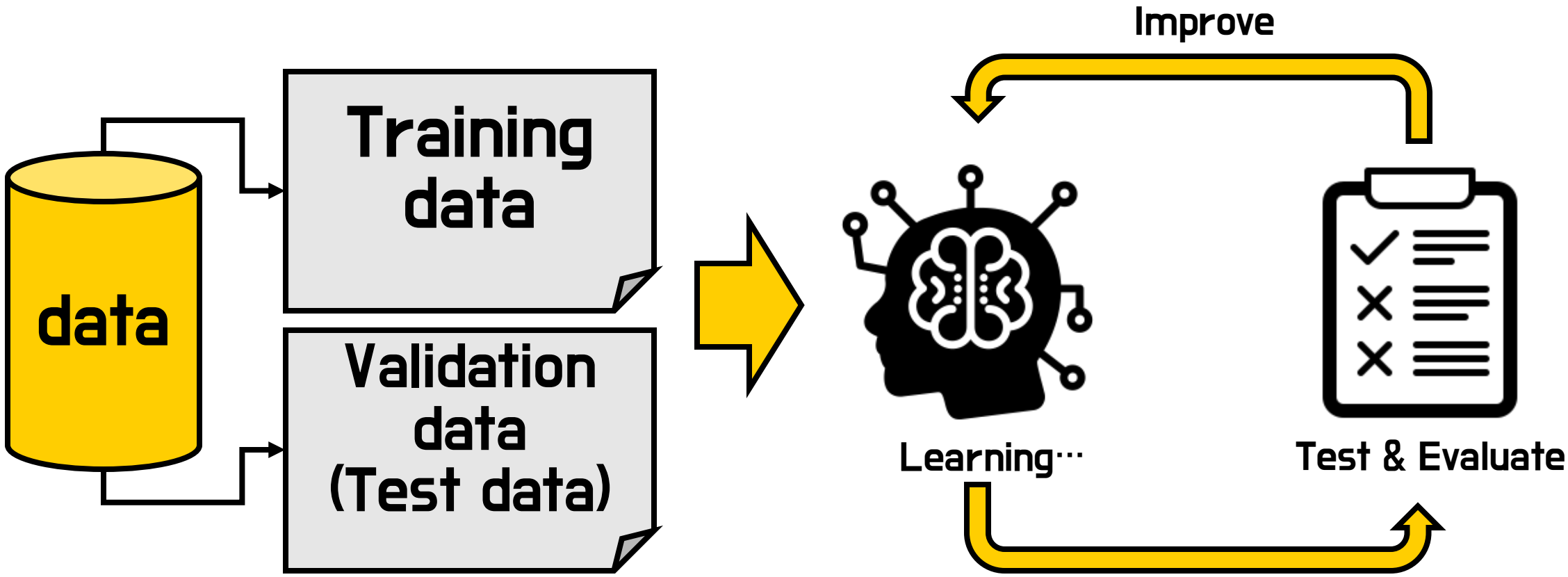


파뿌리
파이썬 뿌시는 이십대들!!

Machine Learning process



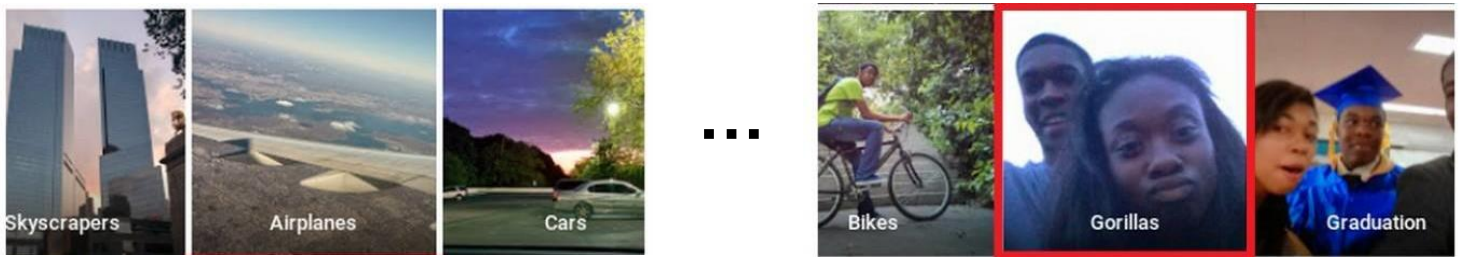
Machine Learning process



파뿌리
파이썬 뿌시는 이십대들!!

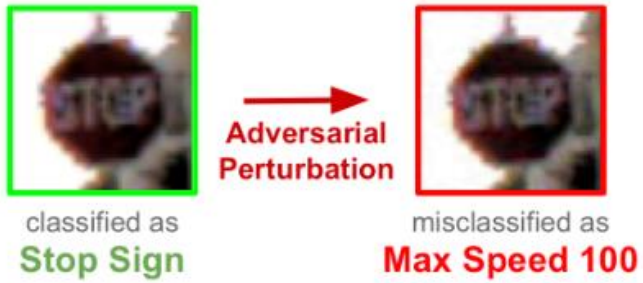
머신러닝은 만능일까?

1. Biased Data를 학습시킬 경우 -> 결과도 biased
(Data의 품질과 양이 중요)



2. 왜 그런 학습결과를 내놓았는지 설명하지 못 한다…
(또, 같은 모델을 학습하더라도 정확도는 다르다.)

3. 보안에 취약하다.



파뿌리
파이썬 뿌시는 이십대들!!

《 Round 6 》

- 머신러닝 개요 - complete
- 단순회귀분석 실습 《
- 다항회귀분석 실습

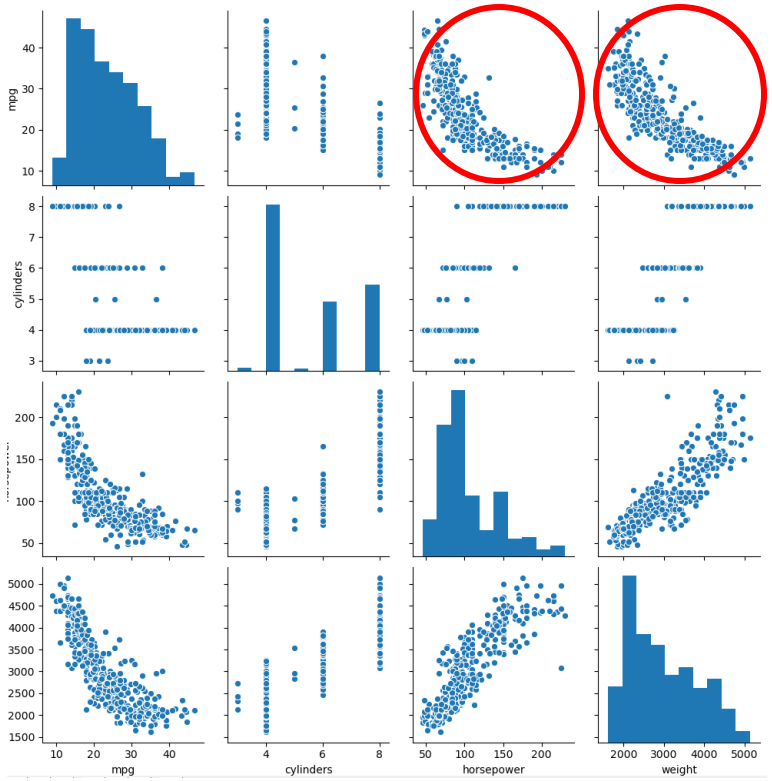
파뿌리
파이썬 뿌시는 이십대들!!

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
pd.set_option('display.expand_frame_repr', False) # DataFrame 출력시 짤림 해결

# seaborn의 mpg 데이터셋 가져오기
mpg_df = sns.load_dataset('mpg')

## 데이터 확인
# 데이터 살펴보기
print(mpg_df.info())
print(mpg_df.head())
print(mpg_df.tail())
print(mpg_df.describe(include='all'))
print('----- 데이터 확인 중 -----\n')


## 출력에 noise 존재 및 출시년도, 제조국 번호가 범주형 아닌 연속형임 확인 -> 전처리
# 출시년도 및 제조국 번호를 astype() 메소드로 int -> object 변환
mpg_df[['model_year', 'origin']] = mpg_df[['model_year', 'origin']].astype('object')
print(mpg_df.describe(include="all"),"\n")

# 결측값 대체 - 기존 DF를 복제하고 '출력 column'에 '?'가 있는 row을 제거하고 출력의 평균을 구해서 hpMean에 삽입
mpg_df_ex = mpg_df
mpg_df_ex['horsepower'].replace('?', np.nan, inplace=True)
mpg_df_ex.dropna(subset=['horsepower'], axis=0, inplace=True)
mpg_df_ex['horsepower'] = mpg_df_ex['horsepower'].astype('float')
hpMean = round(mpg_df_ex['horsepower'].mean(),3)
print("결측값용 대체값 :",hpMean,"\n")

# 결측값 대체 - ?를 제외한 출력의 평균을 결측치들에 넣어준다.
mpg_df_ex['horsepower'].replace('?', hpMean, inplace=True)
mpg_df_ex.dropna(subset=['horsepower'], axis=0, inplace=True)
mpg_df_ex['horsepower'] = mpg_df_ex['horsepower'].astype('float')

print(mpg_df.describe(include='all'))
print('----- 데이터 전처리 완료 -----\n')
```

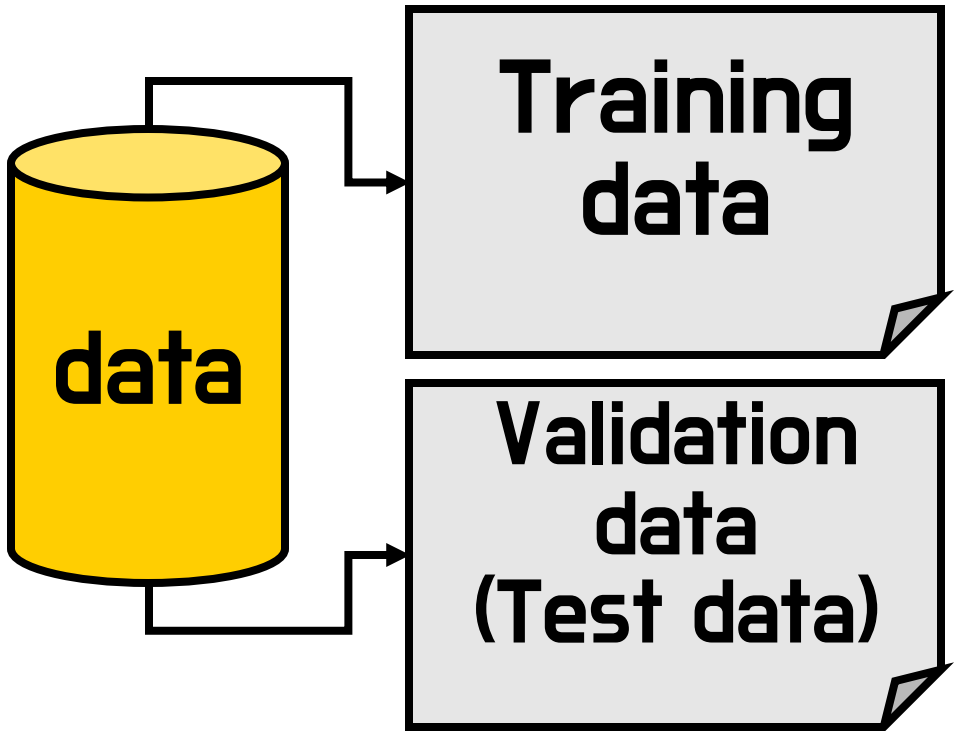
source code :

https://github.com/KGJsGit/Python_Breakers/blob/master/source_code/simLinReg.py

```
## 변수 선택
# 분석에 활용할 변수 선택 (연비, 실린더, 출력, 중량)
ndf = mpg_df[['mpg', 'cylinders', 'horsepower', 'weight']]

# 두 변수씩 짝을 지을 수 있는 모든 경우에 대한 그래프(산점도) 확인(seaborn.pairplot())
sns.pairplot(ndf)
plt.show()
plt.close()
```

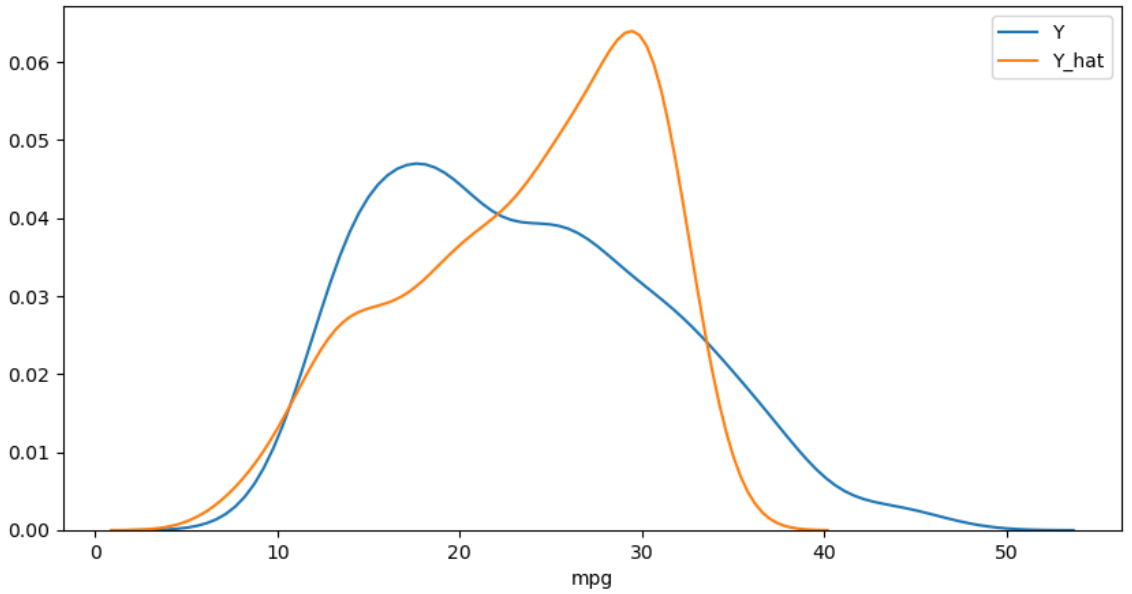


```
## mpg와 horsepower, weight가 선형관계임이 확인 weight을 x, mpg를 y로 선택
# 속성(변수) 선택
X = ndf[['weight']]  #독립 변수 X
Y = ndf['mpg']       #종속 변수 Y


## dataset을 training data와 test data로 분할
# train_test_split(독립변수, 종속변수, test data 사이즈(%), 랜덤 추출 시드값)
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3, random_state=10)

print('train data 개수: ', len(X_train))
print('test data 개수: ', len(X_test),"\n")
```



```python
# sklearn 라이브러리에서 선형회귀분석 모듈 가져오기
from sklearn.linear_model import LinearRegression

# 단순회귀분석 모형 객체 생성
lr = LinearRegression()

## 학습 시작
# train data를 가지고 모형 학습
lr.fit(X_train, Y_train)

# 학습을 마친 모형에 test data를 적용하여 결정계수(R^2) 계산
r_square = lr.score(X_test, Y_test)

# 회귀식과 결정계수(R^2) 산출
print('회귀식 :', float(lr.coef_),'X +',lr.intercept_)
print('결정계수(R^2) :', r_square)
print('\n')

# 모형에 전체 X 데이터를 입력하여 예측한 값 y_hat을 실제 값 y와 비교
Y_hat = lr.predict(X)

plt.figure(figsize=(10, 5))
ax1 = sns.distplot(Y, hist=False, label="Y")
ax2 = sns.distplot(Y_hat, hist=False, label="Y_hat", ax=ax1)
plt.show()
plt.close()
```

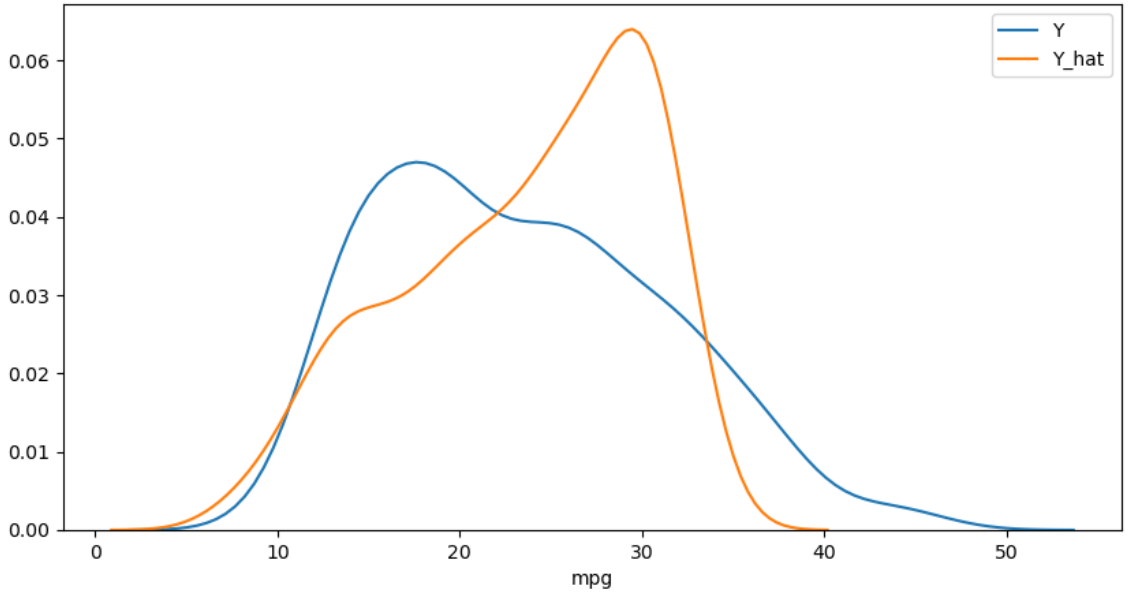
```
train data 개수:  274
test data 개수:  118

회귀식 : -0.007753431671236769 X + 46.7103662572801
결정계수(R^2) : 0.6822458558299325
```



파뿌리
파이썬 뿌시는 이십대들!!

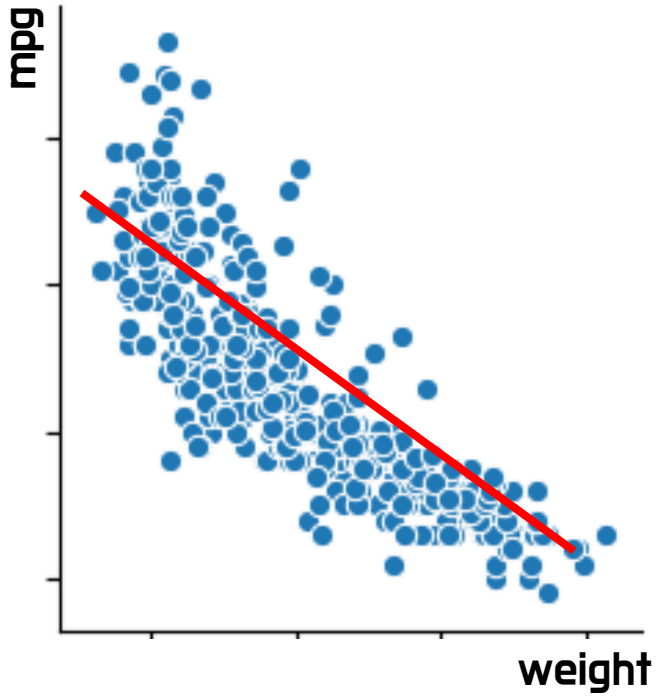
《 Round 6 》

- 머신러닝 개요 - complete
- 단순회귀분석 실습 - complete
- 다항회귀분석 실습 《



뭐야??? 머신러닝 좋다면서!!

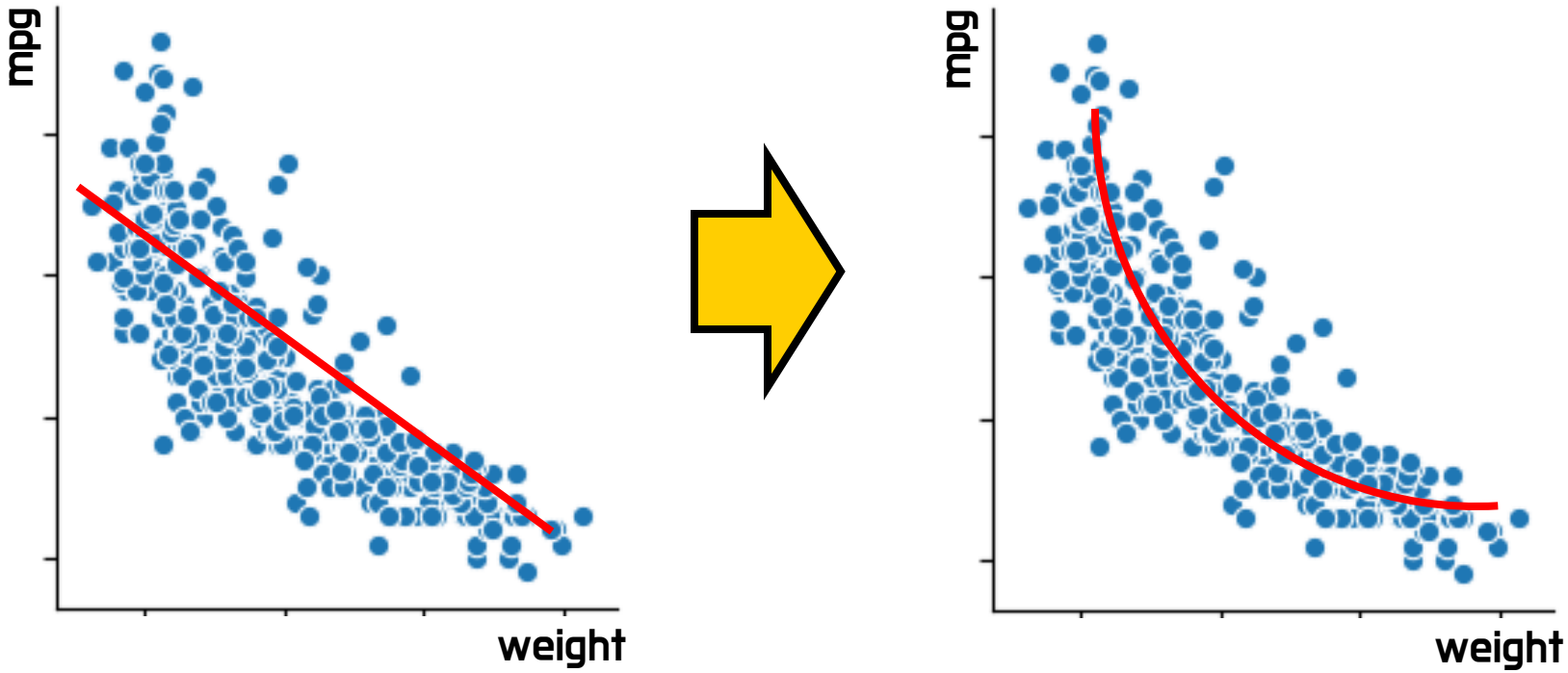
단순회귀분석은 정말 단순하다



$Y = \beta 0 + \beta 1X$

두 변수 간의 관계를 직선으로 설명했기 때문에 정확도가 낮을 수 밖에 없다.

다항회귀분석



$Y = \beta 0 + \beta 1X$

$Y = \beta 0 + \beta 1X^2 + B3X$

두 변수 간의 관계를 곡선으로 설명한다면 정확도를 높일 수 있다!!

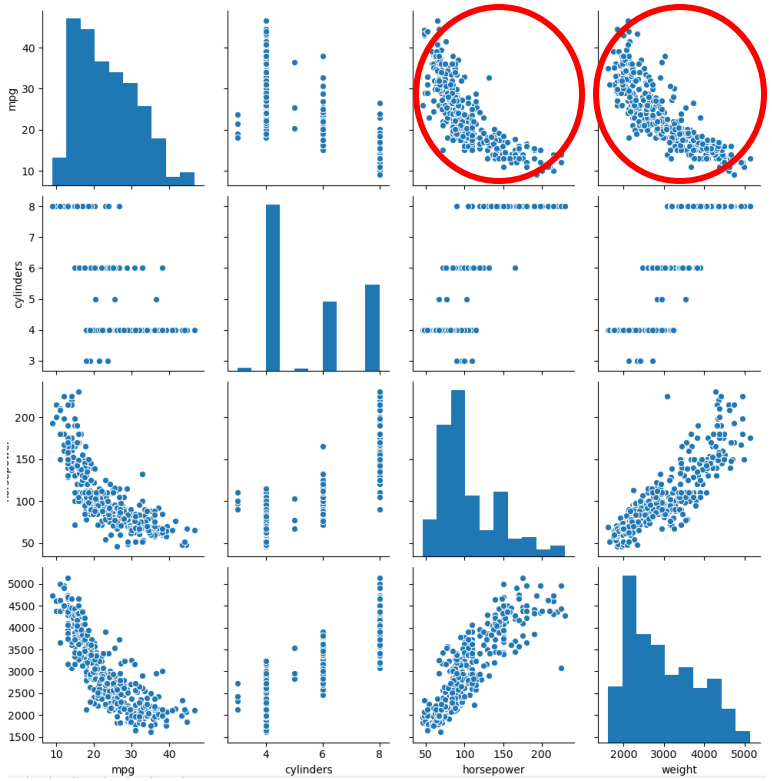
파이썬 뿌시는 이십대들!!

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
pd.set_option('display.expand_frame_repr', False) # DataFrame 출력시 짤림 해결

# seaborn의 mpg 데이터셋 가져오기
mpg_df = sns.load_dataset('mpg')

## 데이터 확인
# 데이터 살펴보기
print(mpg_df.info())
print(mpg_df.head())
print(mpg_df.tail())
print(mpg_df.describe(include='all'))
print('----- 데이터 확인 중 -----\n')


## 출력에 noise 존재 및 출시년도, 제조국 번호가 범주형 아닌 연속형임 확인 -> 전처리
# 출시년도 및 제조국 번호를 astype() 메소드로 int -> object 변환
mpg_df[['model_year', 'origin']] = mpg_df[['model_year', 'origin']].astype('object')
print(mpg_df.describe(include="all"),"\n")

# 결측값 대체 - 기존 DF를 복제하고 '출력 column'에 '?'가 있는 row을 제거하고 출력의 평균을 구해서 hpMean에 삽입
mpg_df_ex = mpg_df
mpg_df_ex['horsepower'].replace('?', np.nan, inplace=True)
mpg_df_ex.dropna(subset=['horsepower'], axis=0, inplace=True)
mpg_df_ex['horsepower'] = mpg_df_ex['horsepower'].astype('float')
hpMean = round(mpg_df_ex['horsepower'].mean(),3)
print("결측값용 대체값 :",hpMean,"\n")

# 결측값 대체 - ?를 제외한 출력의 평균을 결측치들에 넣어준다.
mpg_df_ex['horsepower'].replace('?', hpMean, inplace=True)
mpg_df_ex.dropna(subset=['horsepower'], axis=0, inplace=True)
mpg_df_ex['horsepower'] = mpg_df_ex['horsepower'].astype('float')

print(mpg_df.describe(include='all'))
print('----- 데이터 전처리 완료 -----\n')
```

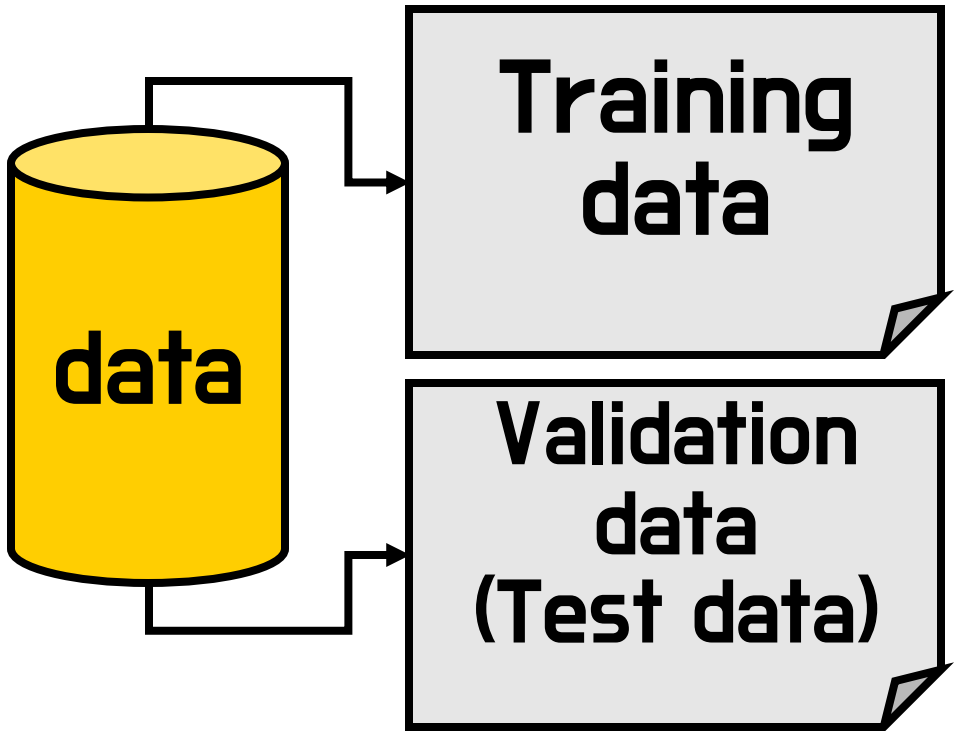
source code :

https://github.com/KGJsGit/Python_Breakers/blob/master/source_code/polLinReg.py

```
## 변수 선택
# 분석에 활용할 변수 선택 (연비, 실린더, 출력, 중량)
ndf = mpg_df[['mpg', 'cylinders', 'horsepower', 'weight']]

# 두 변수씩 짝을 지을 수 있는 모든 경우에 대한 그래프(산점도) 확인(seaborn.pairplot())
sns.pairplot(ndf)
plt.show()
plt.close()
```

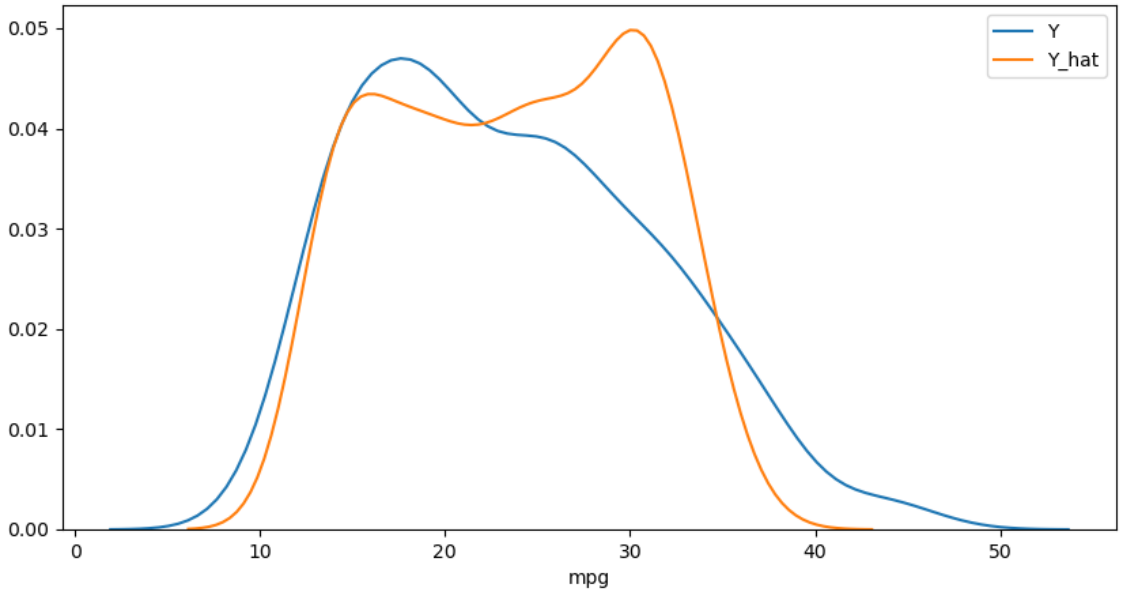


```
## mpg와 horsepower, weight가 선형관계임이 확인 weight을 x, mpg를 y로 선택
# 속성(변수) 선택
X = ndf[['weight']]  #독립 변수 X
Y = ndf['mpg']       #종속 변수 Y


## dataset을 training data와 test data로 분할
# train_test_split(독립변수, 종속변수, test data 사이즈(%), 랜덤 추출 시드값)
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3, random_state=10)

print('train data 개수: ', len(X_train))
print('test data 개수: ', len(X_test),"\n")
```

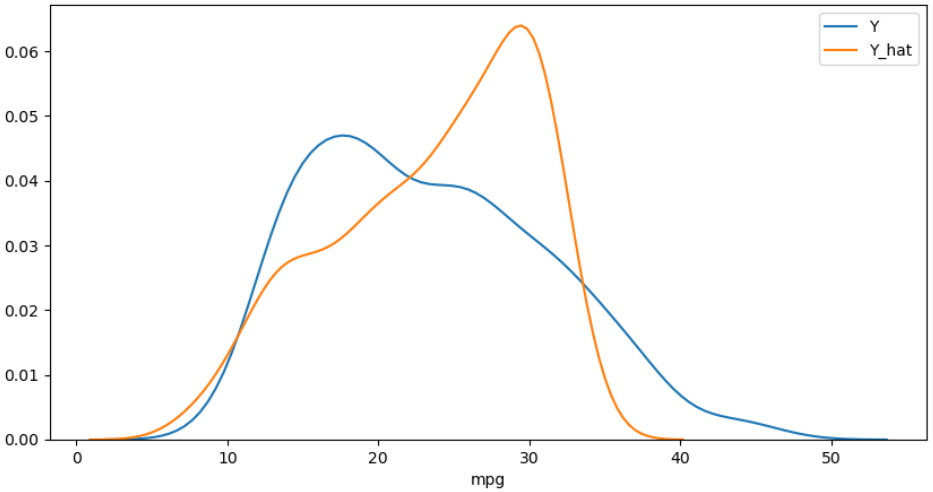


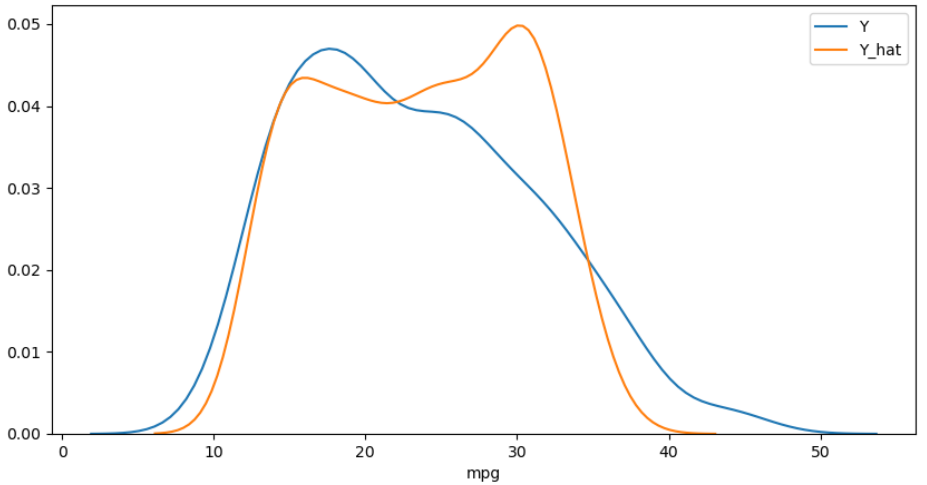
```python
## 2차 방정식의 형태로 변환
# sklearn 라이브러리에서 선형회귀분석 및 다항식 변환 모듈 가져오기
from sklearn.linear_model import LinearRegression #선형회귀분석
from sklearn.preprocessing import PolynomialFeatures #다항식 변환

# 다항식 변환
poly = PolynomialFeatures(degree = 2) #2차항 적용
X_train_poly = poly.fit_transform(X_train) #X_train 데이터를 2차항으로 변형

## 학습 시작
# train data를 가지고 모형 학습
pr = LinearRegression()
pr.fit(X_train_poly, Y_train)

# 학습을 마친 모형에 test data를 적용하여 결정계수(R^2) 계산
X_test_poly = poly.fit_transform(X_test)# X_test 데이터를 2차항으로 변형
r_square = pr.score(X_test_poly,Y_test)
print('회귀식 :', pr.coef_[1],'X^2 +',pr.coef_[2],'X +',pr.intercept_)
print('결정계수(R^2) :', r_square)
print('\n')

# 모형에 전체 X 데이터를 입력하여 예측한 값 y_hat을 실제 값 y와 비교
X_ploy = poly.fit_transform(X)
Y_hat = pr.predict(X_ploy)

plt.figure(figsize = (10, 5))
ax1 = sns.distplot(Y, hist = False, label = "Y")
ax2 = sns.distplot(Y_hat, hist = False, label = "Y_hat", ax = ax1)
plt.show()
plt.close()
```

```
train data 개수:  274
validation data 개수:  118

회귀식 : -0.018576828851341987 X^2 + 1.7049122315940246e-06 X + 62.580712215769495
결정계수(R^2) : 0.7087009262975481
```



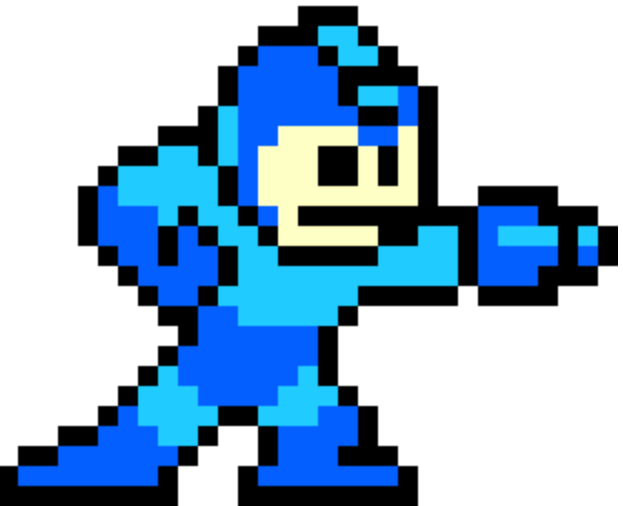
단순회귀분석과 다항회귀분석 결과비교





파뿌리
파이썬 뿌시는 이십대들!!

WARNING

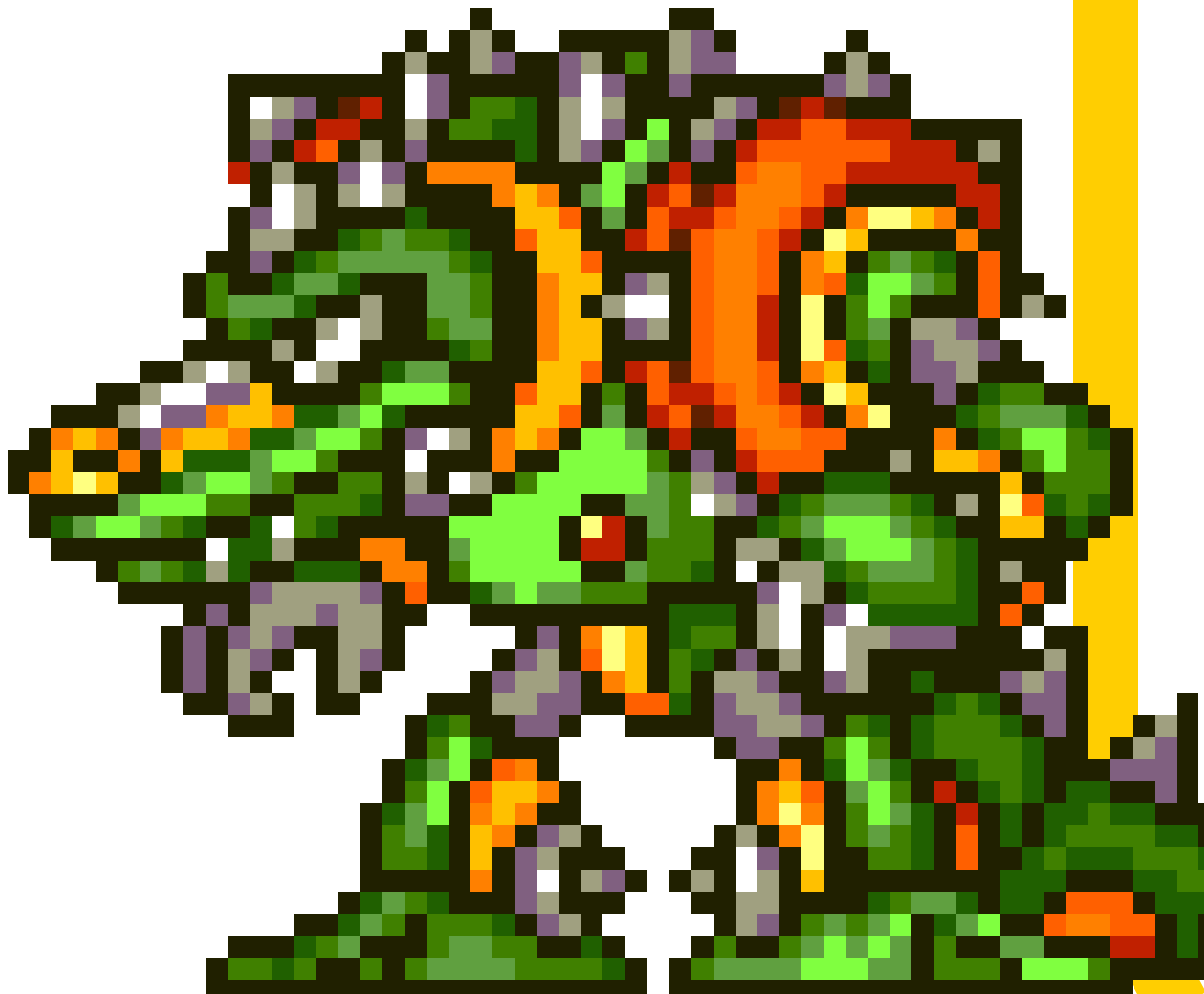


파이썬 뿌시는 이십대들!!

WARNING

《 QUEST 》

지금까지의 데이터분석 과정 복습



파뿌리
파이썬 뿌시는 이십대들!!

NEXT STAGE