고려대학교 의료정보학과

IMDL
Intelligent Medical Data Lab

의료인공지능
머신러닝 - 선형회귀

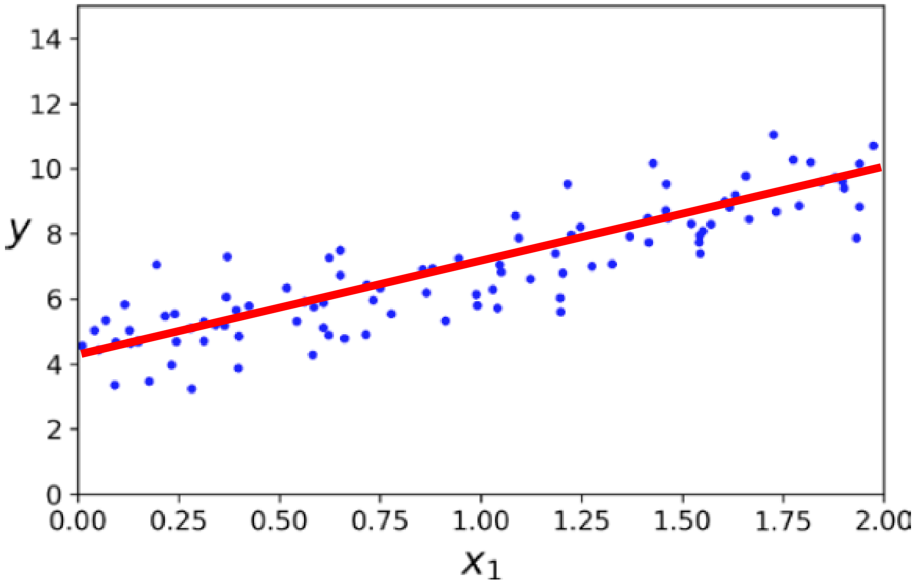
고려대학교 의료빅데이터연구소
채민수(minsuchae@korea.ac.kr)

1. 선형 회귀

◦ 선형회귀

- 최소한의 오차를 갖는 방정식을 찾는 것
- 오차 측정 방법
  - RMSE $$RMSE = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(y_i - h(x)_i)^2}$$
  - MAE $$MAE = \frac{1}{m}\sum_{i=1}^{m}|y_i - h(x)_i|$$
- 선형 회귀 계산 방법

$$\begin{aligned} h(x) &= \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n \\ &= \theta X \end{aligned}$$



1. 선형 회귀

- 단항 회귀
  - 하나의 특징만으로 계산

- 다항 회귀
  - 하나의 특징을 통해 새로운 특징을 도출하거나, 여러 특징으로 계산

IMDL Intelligent Medical Data Lab

2. 단항 회귀 실습

- 데이터셋 구축
  - x : 1 ~ 1000
  - outcome : 3x+5

```
import numpy as np

my_input = np.arange(1,1001)
outcome = []

for x in my_input:
    outcome.append(3*x + 5)
```

- 훈련 데이터와 테스트 데이터로 나누기

```
my_input = my_input.reshape(-1, 1)
from sklearn.model_selection import train_test_split

train_input, test_input, train_target, test_target = train_test_split(my_input, outcome, test_size=0.3, random_state=42)
```

2. 단항 회귀 실습

◦ 선형 회귀 알고리즘으로 학습

```
from sklearn.linear_model import LinearRegression

lr = LinearRegression()
lr.fit(train_input, train_target)
```

◦ 오차 확인

```
y_pred = lr.predict(test_input)
from sklearn.metrics import mean_absolute_error, mean_squared_error

print("MAE :",mean_absolute_error(test_target, y_pred))
print("RMSE :",np.sqrt(mean_squared_error(test_target, y_pred)))
```

```
[실행결과]
MAE : 3.2180480502574937e-13
RMSE : 4.747199243903212e-13
```

IMDL Intelligent Medical Data Lab

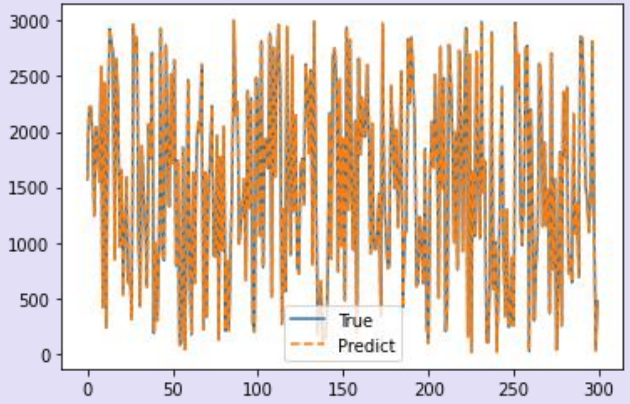
2. 단항 회귀 실습

- 시각화하여 성능 확인

```
%matplotlib inline
import matplotlib.pyplot as plt

plt.plot(test_target, linestyle='-',label='True')
plt.plot(y_pred, linestyle='--', label='Predict')
plt.legend()
plt.show()
```

[실행결과]



2. 단항 회귀 실습

◦ 학습된 가중치 확인(편향와 절편)

```
print(lr.coef_)
print(lr.intercept_)
```

```
[실행결과]
[3.]
4.999999999999773
```

```
for x in my_input:
    outcome.append(3*x + 5)
```

IMDL
Intelligent Medical Data Lab

2. 단항 회귀 실습

- 데이터셋 구축
  - x : 1 ~ 1000
  - outcome : 3x+5+noise

```
import numpy as np

my_input = np.arange(1,1001)
outcome = []

for x in my_input:
    outcome.append(3*x + 5 + np.random.randint(-6,7))
```

- 훈련 데이터와 테스트 데이터로 나누기

```
my_input = my_input.reshape(-1, 1)
from sklearn.model_selection import train_test_split

train_input, test_input, train_target, test_target = train_test_split(my_input, outcome, test_size=0.3,
random_state=42)
```

2. 단항 회귀 실습

- 선형 회귀 알고리즘으로 학습

```
from sklearn.linear_model import LinearRegression

lr = LinearRegression()
lr.fit(train_input, train_target)
```

- 오차 확인

```
y_pred = lr.predict(test_input)
from sklearn.metrics import mean_absolute_error, mean_squared_error

print("MAE :",mean_absolute_error(test_target, y_pred))
print("RMSE :",np.sqrt(mean_squared_error(test_target, y_pred)))
```

```
[실행결과]
MAE : 3.2977564133595845
RMSE : 3.718191148822024
```

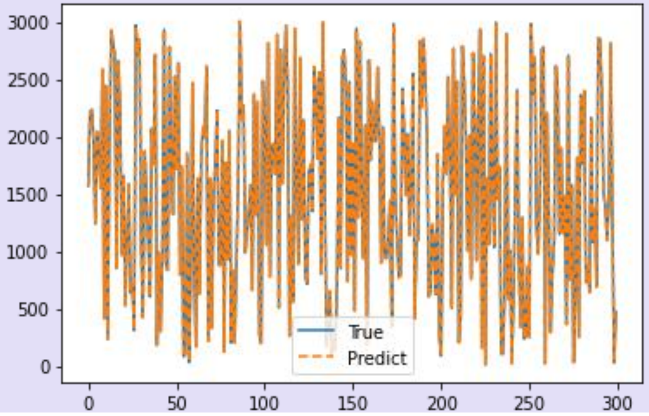
2. 단항 회귀 실습

- 시각화하여 성능 확인

```
%matplotlib inline
import matplotlib.pyplot as plt

plt.plot(test_target, linestyle='-',label='True')
plt.plot(y_pred, linestyle='--', label='Predict')
plt.legend()
plt.show()
```

[실행결과]



2. 단항 회귀 실습

◦ 학습된 가중치 확인(편향와 절편)

```
print(lr.coef_)
print(lr.intercept_)
```

```
[실행결과]
[3.00055562]
4.44316208062105
```

```
for x in my_input:
    outcome.append(3*x + 5 + np.random.randint(-6,7))
```

3. 다항 회귀 실습

- 데이터셋 구축
  - x : 1 ~ 1000
  - outcome : $3x^2+5x+5$ + noise

```
import numpy as np

my_input = np.arange(1,1001)
outcome = []

for x in my_input:
    outcome.append(3*x*x + 5*x + 5 + np.random.randint(-6,7))
```

- 훈련 데이터와 테스트 데이터로 나누기

```
my_input = my_input.reshape(-1, 1)
from sklearn.model_selection import train_test_split

train_input, test_input, train_target, test_target = train_test_split(my_input, outcome, test_size=0.3, random_state=42)
```

IMDL Intelligent Medical Data Lab

3. 다항 회귀 실습

- 선형 회귀 알고리즘으로 학습

```
from sklearn.linear_model import LinearRegression

lr = LinearRegression()
lr.fit(train_input, train_target)
```

- 오차 확인

```
y_pred = lr.predict(test_input)
from sklearn.metrics import mean_absolute_error, mean_squared_error

print("MAE :",mean_absolute_error(test_target, y_pred))
print("RMSE :",np.sqrt(mean_squared_error(test_target, y_pred)))
```

```
[실행결과]
MAE : 198154.35217320398
RMSE : 230166.8283753354
```

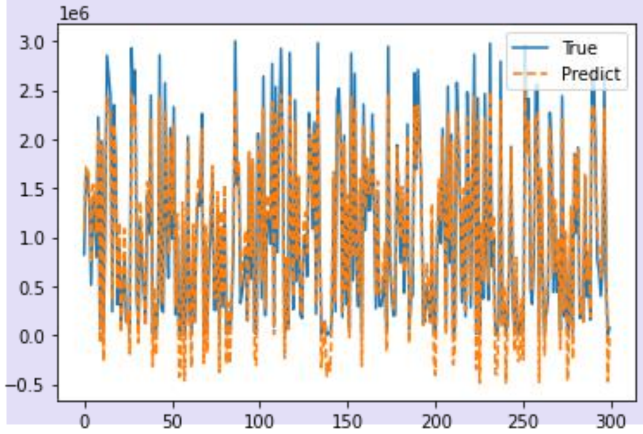
3. 다항 회귀 실습

◦ 시각화하여 성능 확인

```
%matplotlib inline
import matplotlib.pyplot as plt

plt.plot(test_target, linestyle='-',label='True')
plt.plot(y_pred, linestyle='--', label='Predict')
plt.legend()
plt.show()
```

[실행결과]



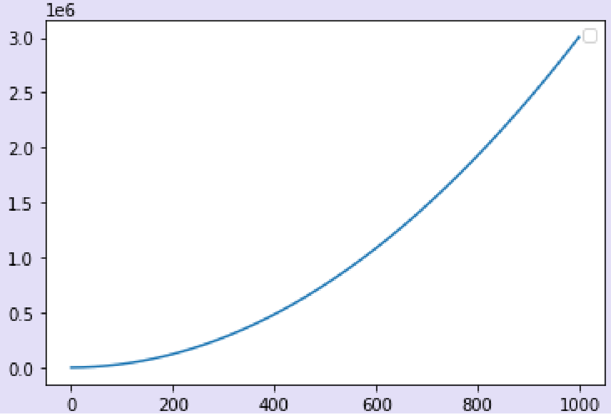
3. 다항 회귀 실습

- 학습이 안된 원인 파악

```
%matplotlib inline
import matplotlib.pyplot as plt

plt.plot(outcome, linestyle='-')
plt.legend()
plt.show()
```

[실행결과]



Intelligent Medical Data Lab

3. 다항 회귀 실습

◦ 기존 특징을 이용하여 새로운 특징 도출

```
my_input = my_input.reshape(-1, 1)
from sklearn.preprocessing import PolynomialFeatures

poly = PolynomialFeatures(degree=2)
my_input_poly = poly.fit_transform(my_input)
```

◦ 훈련 데이터와 테스트 데이터 나누기

```
from sklearn.model_selection import train_test_split

train_input, test_input, train_target, test_target = train_test_split(my_input_poly, 
outcome, test_size=0.3, random_state=42)
```

3. 다항 회귀 실습

◦ 선형 회귀 알고리즘으로 학습

```
from sklearn.linear_model import LinearRegression

lr = LinearRegression()
lr.fit(train_input, train_target)
```

◦ 오차 확인

```
y_pred = lr.predict(test_input)
from sklearn.metrics import mean_absolute_error, mean_squared_error

print("MAE :",mean_absolute_error(test_target, y_pred))
print("RMSE :",np.sqrt(mean_squared_error(test_target, y_pred)))
```

```
[실행결과]
MAE : 3.1817412657126054
RMSE : 3.675293210723283
```

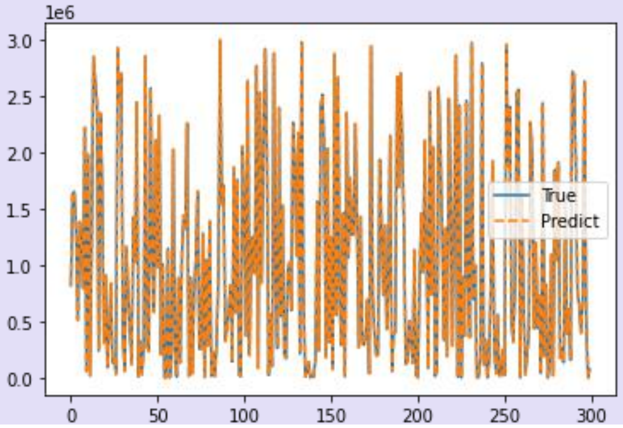
3. 다항 회귀 실습

◦ 시각화하여 성능 확인

```
%matplotlib inline
import matplotlib.pyplot as plt

plt.plot(test_target, linestyle='-',label='True')
plt.plot(y_pred, linestyle='--', label='Predict')
plt.legend()
plt.show()
```

[실행결과]



3. 다항 회귀 실습

◦ 학습된 가중치 확인(편향와 절편)

```
print(lr.coef_)
print(lr.intercept_)
```

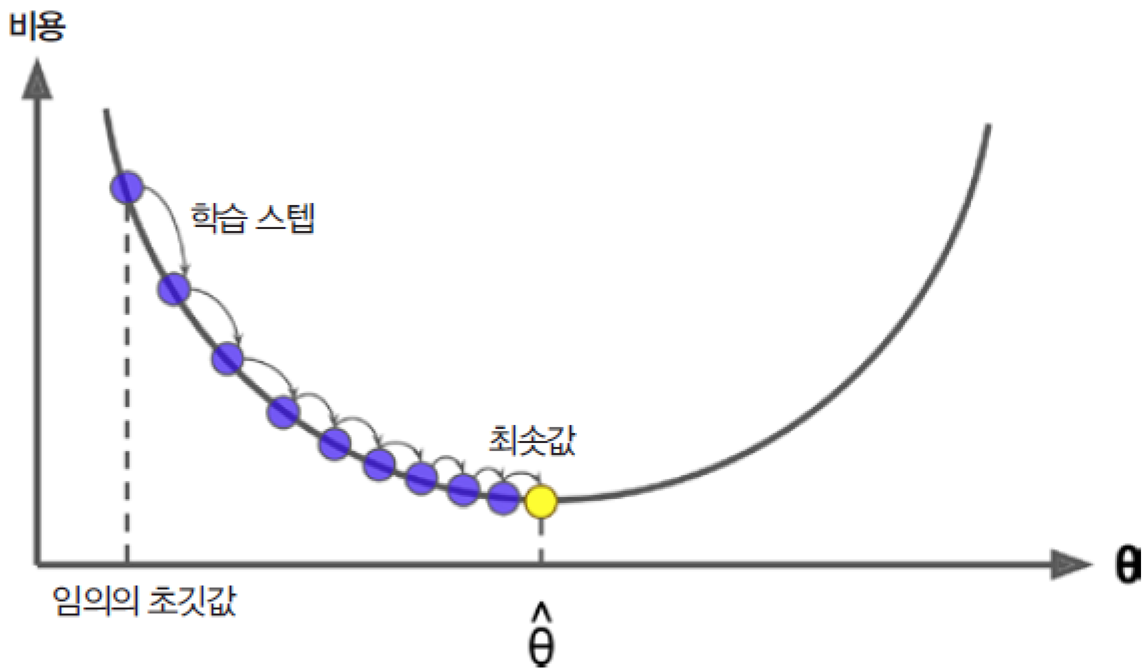
```
[실행결과]
[0.         4.99901883 3.00000082]
5.3169000196503475
```

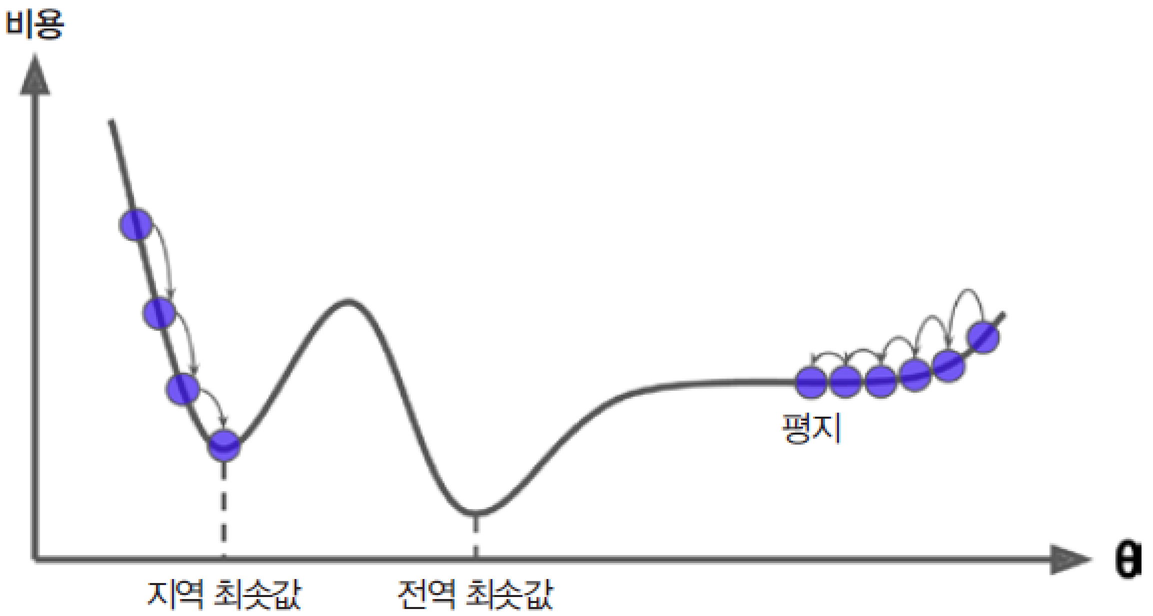
```
for x in my_input:
    outcome.append(3*x*x + 5*x + 5 + np.random.randint(-6,7))
```

IMDL
Intelligent Medical Data Lab

4. 경사하강법

◦ 경사하강법(Gradient decent,GD)





- 학습률 : 한 번에 학습하는 데이터의 수

4. 경사하강법

◦ 배치 경사하강법

- 비용 함수의 편도 함수

$$\frac{\partial}{\partial \theta_j} MSE(\theta) = \frac{2}{m} \sum_{i=1}^{m} \left(\theta^T x^i - y^i\right) x_j^i$$

- 비용 함수의 그레이디언트 벡터

$$\nabla_\theta MSE(\theta) = \begin{pmatrix} \frac{\partial}{\partial \theta_0} MSE(\theta) \\ \frac{\partial}{\partial \theta_1} MSE(\theta) \\ \vdots \\ \frac{\partial}{\partial \theta_n} MSE(\theta) \end{pmatrix} = \frac{2}{m} X^T (X\theta - y)$$

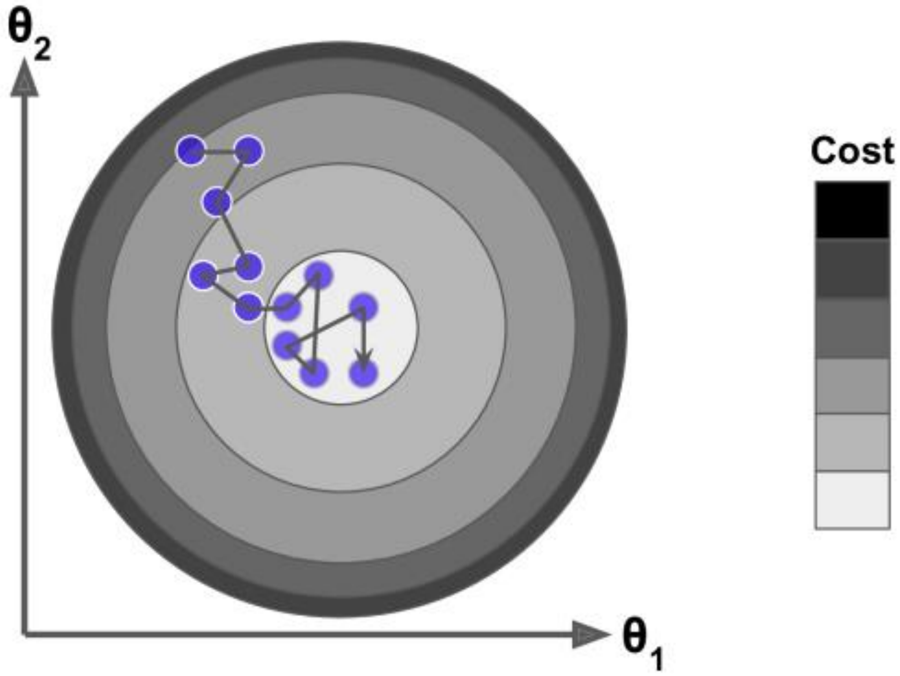
- 경사하강법 스텝

$$\theta^{(\text{next step})} = \theta - \eta \nabla_\theta MSE(\theta)$$

4. 경사하강법

◦ 확률적 경사하강법

- 전체의 데이터가 아닌 무작위 데이터를 선택하여 이를 바탕으로 계산



5. 경사하강법 실습

- 데이터셋 로드

```
import pandas as pd

df = pd.read_csv('Medical_Insurance_dataset.csv')
df.head()
```

[실행결과]

	age	sex	bmi	smoker	region	children	charges
0	21.000000	male	25.745000	no	northeast	2	3279.868550
1	36.976978	female	25.744165	yes	southeast	3	21454.494239
2	18.000000	male	30.030000	no	southeast	1	1720.353700
3	37.000000	male	30.676891	no	northeast	3	6801.437542
4	58.000000	male	32.010000	no	southeast	1	11946.625900

5. 경사하강법 실습

◦ 데이터 전처리

```
df.loc[df['sex']=='male','sex']=0
df.loc[df['sex']=='female','sex']=1
df['sex']=df['sex'].astype('int32')

df.loc[df['smoker']=='no','smoker']=0
df.loc[df['smoker']=='yes','smoker']=1
df['smoker']=df['smoker'].astype('int32')

df = pd.get_dummies(df)
df.head()
```

[실행결과]

	age	sex	bmi	smoker	children	charges	region_northeast	region_northwest	region_southeast	region_southwest
0	21.000000	0	25.745000	0	2	3279.868550	1	0	0	0
1	36.976978	1	25.744165	1	3	21454.494239	0	0	1	0
2	18.000000	0	30.030000	0	1	1720.353700	0	0	1	0
3	37.000000	0	30.676891	0	3	6801.437542	1	0	0	0
4	58.000000	0	32.010000	0	1	11946.625900	0	0	1	0

5. 경사하강법 실습

◦ 데이터 범위 확인

```
df.describe()
```

[실행결과]

	age	sex	bmi	smoker	children	charges	region_northeast	region_northwest	region_southeast	region_southwe
count	3630.000000	3630.000000	3630.000000	3630.000000	3630.000000	3630.000000	3630.000000	3630.000000	3630.000000	3630.0000
mean	38.887036	0.441047	30.629652	0.154270	2.503581	12784.808644	0.233609	0.250964	0.281267	0.2341
std	12.151029	0.496581	5.441307	0.361257	1.712568	10746.166743	0.423184	0.433628	0.449680	0.4235
min	18.000000	0.000000	15.960000	0.000000	0.000000	1121.873900	0.000000	0.000000	0.000000	0.0000
25%	29.000000	0.000000	26.694526	0.000000	1.000000	5654.818262	0.000000	0.000000	0.000000	0.0000
50%	39.170922	0.000000	30.200000	0.000000	3.000000	9443.807222	0.000000	0.000000	0.000000	0.0000
75%	48.343281	1.000000	34.100000	0.000000	4.000000	14680.407505	0.000000	1.000000	1.000000	0.0000
max	64.000000	1.000000	53.130000	1.000000	5.000000	63770.428010	1.000000	1.000000	1.000000	1.0000

5. 경사하강법 실습

◦ 훈련 데이터와 테스트 데이터 나누기

```
feature = df[df.keys().drop('charges')].values
outcome = df['charges'].values

from sklearn.model_selection import train_test_split

train_input, test_input, train_target, test_target = train_test_split(feature,
outcome, test_size=0.3, random_state=42)
```

◦ 사이킷런에서 제공하는 확률적 경사하강법 학습

```
from sklearn.linear_model import SGDRegressor

sgd = SGDRegressor()
sgd.fit(train_input, train_target)
```

IMDL Intelligent Medical Data Lab

5. 경사하강법 실습

◦ 오차 확인

```
y_pred = sgd.predict(test_input)
import numpy as np
from sklearn.metrics import mean_absolute_error, mean_squared_error

print("MAE :",mean_absolute_error(test_target, y_pred))
print("RMSE :",np.sqrt(mean_squared_error(test_target, y_pred)))
```

```
[실행결과]
MAE : 5517.404681501185
RMSE : 7544.05451163124
```

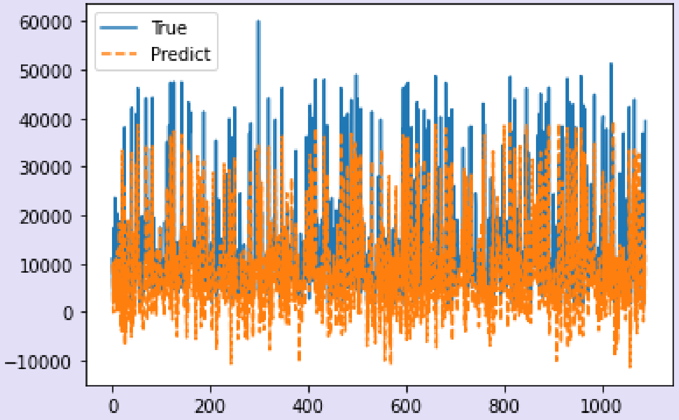
5. 경사하강법 실습

◦ 시각화

```
%matplotlib inline
import matplotlib.pyplot as plt

plt.plot(test_target, linestyle='-',label='True')
plt.plot(y_pred, linestyle='--', label='Predict')
plt.legend()
plt.show()
```

[실행결과]



5. 경사하강법 실습

◦ 훈련 데이터와 테스트 데이터 나누기

```
feature = df[df.keys().drop('charges')].values
outcome = df['charges'].values

from sklearn.model_selection import train_test_split

train_input, test_input, train_target, test_target = train_test_split(feature,
outcome, test_size=0.3, random_state=42)
```

◦ 데이터 스케일 수행

```
from sklearn.preprocessing import MinMaxScaler

feature_scaler = MinMaxScaler()
train_input_scaled = feature_scaler.fit_transform(train_input)
test_input_scaled = feature_scaler.transform(test_input)
```

5. 경사하강법 실습

◦ 사이킷런에서 제공하는 확률적 경사하강법 학습

```
from sklearn.linear_model import SGDRegressor

sgd = SGDRegressor()
sgd.fit(train_input_scaled, train_target)
```

◦ 오차 확인

```
y_pred = sgd.predict(test_input_scaled)
from sklearn.metrics import mean_absolute_error, mean_squared_error

print("MAE :",mean_absolute_error(test_target, y_pred))
print("RMSE :",np.sqrt(mean_squared_error(test_target, y_pred)))
```

```
[실행결과]
MAE : 3862.0216205787115
RMSE : 5799.596912892567
```

IMDL Intelligent Medical Data Lab

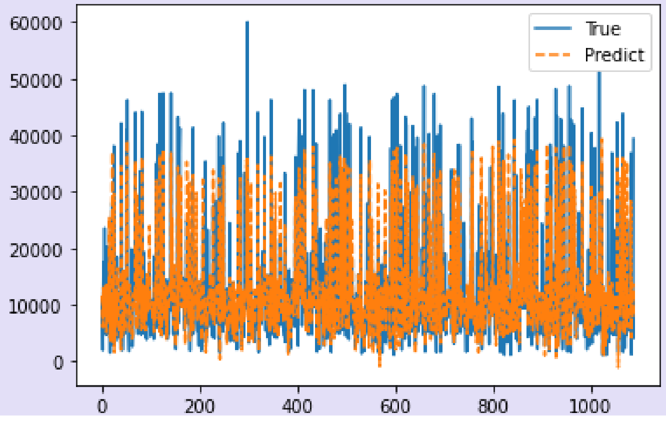
5. 경사하강법 실습

- 시각화

```
%matplotlib inline
import matplotlib.pyplot as plt

plt.plot(test_target, linestyle='-',label='True')
plt.plot(y_pred, linestyle='--', label='Predict')
plt.legend()
plt.show()
```

[실행결과]



5. 경사하강법 실습

- 기존 특징을 이용하여 새로운 특징 도출

```
from sklearn.preprocessing import PolynomialFeatures

poly = PolynomialFeatures(degree=2)
feature_poly = poly.fit_transform(feature)
```

- 훈련 데이터와 테스트 데이터 나누기

```
from sklearn.model_selection import train_test_split

train_input, test_input, train_target, test_target = train_test_split(feature_poly , outcome, test_size=0.3, random_state=42)
```

5. 경사하강법 실습

◦ 데이터 스케일 수행

```
from sklearn.preprocessing import MinMaxScaler

feature_scaler = MinMaxScaler()
train_input_scaled = feature_scaler.fit_transform(train_input)
test_input_scaled = feature_scaler.transform(test_input)
```

◦ 사이킷런에서 제공하는 확률적 경사하강법 학습

```
from sklearn.linear_model import SGDRegressor

sgd = SGDRegressor()
sgd.fit(train_input_scaled, train_target)
```

5. 경사하강법 실습

◦ 오차 확인

```
y_pred = sgd.predict(test_input_scaled)
from sklearn.metrics import mean_absolute_error, mean_squared_error

print("MAE :",mean_absolute_error(test_target, y_pred))
print("RMSE :",np.sqrt(mean_squared_error(test_target, y_pred)))
```

```
[실행결과]
MAE : 2980.191767644841
RMSE : 4973.513624218002
```

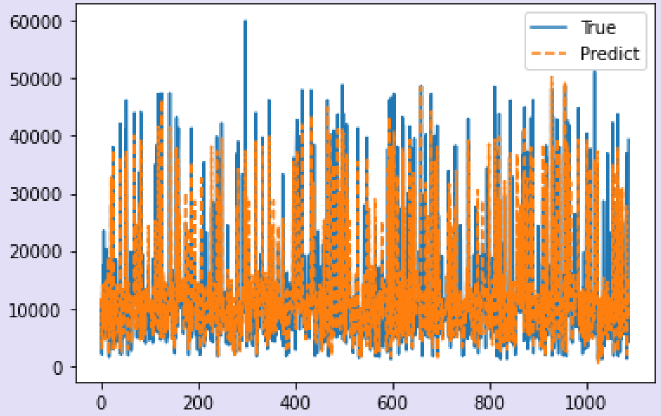
5. 경사하강법 실습

◦ 시각화

```
%matplotlib inline
import matplotlib.pyplot as plt

plt.plot(test_target, linestyle='-',label='True')
plt.plot(y_pred, linestyle='--', label='Predict')
plt.legend()
plt.show()
```

[실행결과]



6. 규제가 있는 선형 회귀

- 릿지 회귀 : 규제항을 통해 가중치가 가능한 작도록 유지(L2 규제)

$$J(\theta) = MSE(\theta) + \alpha \frac{1}{m} \sum_{i=1}^{m} {\theta_i}^2$$

$$\|y - Xw\|_2^2 + \alpha \|w\|_2^2$$

- 라쏘 회귀 : 덜 중요한 가중치를 제거하도록 함(L1 규제)

$$J(\theta) = MSE(\theta) + \alpha \sum_{i=1}^{m} |\theta_i|$$

$$\frac{1}{2m} \|y - Xw\|_2^2 + \alpha \|w\|_1$$

- 엘라스틱넷 : 릿지 회귀와 라쏘 회귀의 절충안

$$J(\theta) = MSE(\theta) + r\alpha \textstyle\sum_{i=1}^{m} |\theta_i| + \frac{1-r}{m} a \textstyle\sum_{i=1}^{m} {\theta_i}^2$$

$$\frac{1}{2m} \|y - Xw\|_2^2 + \alpha \times l1\_latio \|w\|_1 + 0.5a \times (1 - l1\_latio) \|w\|_2^2$$

7. 규제가 있는 선형 회귀 실습 - 릿지

◦ 훈련 데이터와 테스트 데이터 나누기

```
feature = df[df.keys().drop('charges')].values
outcome = df['charges'].values

from sklearn.model_selection import train_test_split

train_input, test_input, train_target, test_target = train_test_split(feature,
outcome, test_size=0.3, random_state=42)
```

◦ 릿지 회귀 학습

```
from sklearn.linear_model import Ridge

ridge = Ridge()
ridge.fit(train_input, train_target)
```

7. 규제가 있는 선형 회귀 실습 - 릿지

- 오차 확인

```
y_pred = ridge.predict(test_input)
import numpy as np
from sklearn.metrics import mean_absolute_error, mean_squared_error

print("MAE :",mean_absolute_error(test_target, y_pred))
print("RMSE :",np.sqrt(mean_squared_error(test_target, y_pred)))
```

```
[실행결과]
MAE : 3851.9773178422897
RMSE : 5809.784337907534
```

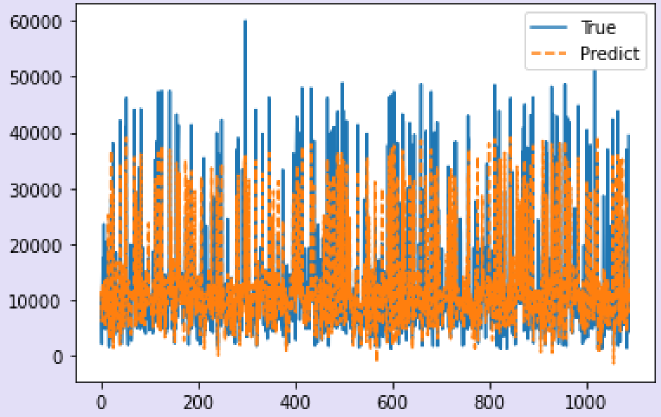
7. 규제가 있는 선형 회귀 실습 - 릿지

- 시각화

```
%matplotlib inline
import matplotlib.pyplot as plt

plt.plot(test_target, linestyle='-',label='True')
plt.plot(y_pred, linestyle='--', label='Predict')
plt.legend()
plt.show()
```

[실행결과]



7. 규제가 있는 선형 회귀 실습 - 릿지

◦ 훈련 데이터와 테스트 데이터 나누기

```
from sklearn.preprocessing import PolynomialFeatures

poly = PolynomialFeatures(degree=2)
feature_poly = poly.fit_transform(feature)

from sklearn.model_selection import train_test_split

train_input, test_input, train_target, test_target = train_test_split(feature_poly, outcome, test_size=0.3, random_state=42)
```

◦ 릿지 회귀 학습

```
from sklearn.linear_model import Ridge

ridge = Ridge()
ridge.fit(train_input, train_target)
```

IMDL Intelligent Medical Data Lab

7. 규제가 있는 선형 회귀 실습 - 릿지

◦ 오차 확인

```
y_pred = ridge.predict(test_input)
import numpy as np
from sklearn.metrics import mean_absolute_error, mean_squared_error

print("MAE :",mean_absolute_error(test_target, y_pred))
print("RMSE :",np.sqrt(mean_squared_error(test_target, y_pred)))
```

```
[실행결과]
MAE : 2778.538043386358
RMSE : 4732.420750666523
```

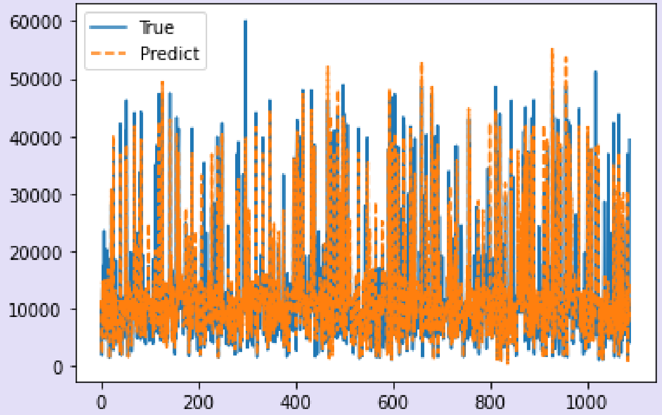
7. 규제가 있는 선형 회귀 실습 - 릿지

◦ 시각화

```
%matplotlib inline
import matplotlib.pyplot as plt

plt.plot(test_target, linestyle='-',label='True')
plt.plot(y_pred, linestyle='--', label='Predict')
plt.legend()
plt.show()
```

[실행결과]



7. 규제가 있는 선형 회귀 실습 - 릿지

- 최적의 alpha 구하기

```
from sklearn.model_selection import train_test_split

train_input, test_input, train_target, test_target = train_test_split(feature,
outcome, test_size=0.3, random_state=42)
```

- 릿지 회귀 학습

```
from sklearn.linear_model import RidgeCV

ridgecv = RidgeCV(alphas=np.arange(0.01,10.01,0.01), cv=5)
ridgecv.fit(train_input, train_target)
```

7. 규제가 있는 선형 회귀 실습 - 릿지

- alpha 확인

```
print(ridgecv.alpha_)
```

```
[실행결과]
0.43
```

- 오차 확인

```
y_pred = ridgecv.predict(test_input)
from sklearn.metrics import mean_absolute_error, mean_squared_error

print("MAE :",mean_absolute_error(test_target, y_pred))
print("RMSE :",np.sqrt(mean_squared_error(test_target, y_pred)))
```

```
[실행결과]
MAE : 3849.0339902851397
RMSE : 5808.61504832412
```

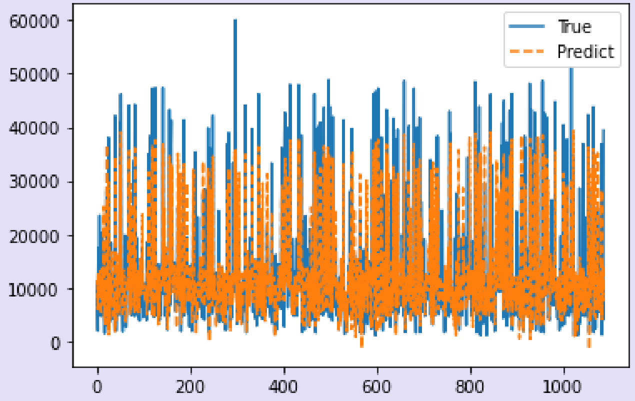
7. 규제가 있는 선형 회귀 실습 - 릿지

◦ 시각화

```
%matplotlib inline
import matplotlib.pyplot as plt

plt.plot(test_target, linestyle='-',label='True')
plt.plot(y_pred, linestyle='--', label='Predict')
plt.legend()
plt.show()
```

[실행결과]



7. 규제가 있는 선형 회귀 실습 - 릿지

◦ 최적의 alpha 구하기 - PolynomialFeatures

```
from sklearn.model_selection import train_test_split

train_input, test_input, train_target, test_target = train_test_split(feature_poly,
outcome, test_size=0.3, random_state=42)
```

◦ 릿지 회귀 학습

```
from sklearn.linear_model import RidgeCV

ridgecv = RidgeCV(alphas=np.arange(0.01,10.01,0.01), cv=5)
ridgecv.fit(train_input, train_target)
```

7. 규제가 있는 선형 회귀 실습 - 릿지

◦ alpha 확인

```
print(ridgecv.alpha_)
```

```
[실행결과]
1.06
```

◦ 오차 확인

```
y_pred = ridgecv.predict(test_input)
from sklearn.metrics import mean_absolute_error, mean_squared_error

print("MAE :",mean_absolute_error(test_target, y_pred))
print("RMSE :",np.sqrt(mean_squared_error(test_target, y_pred)))
```

```
[실행결과]
MAE : 2778.540405036029
RMSE : 4732.537202554494
```

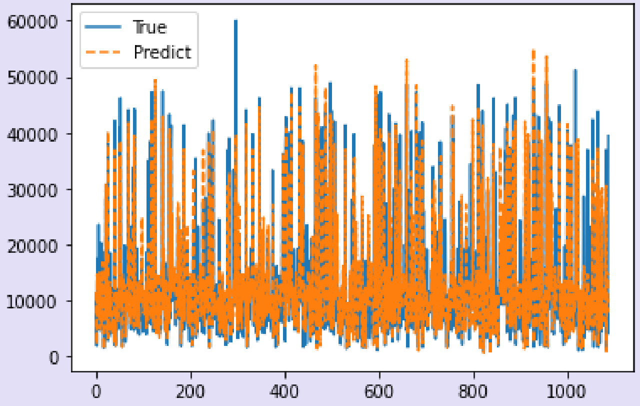
7. 규제가 있는 선형 회귀 실습 - 릿지

◦ 시각화

```
%matplotlib inline
import matplotlib.pyplot as plt

plt.plot(test_target, linestyle='-',label='True')
plt.plot(y_pred, linestyle='--', label='Predict')
plt.legend()
plt.show()
```

[실행결과]



IMDL
Intelligent Medical Data Lab

7. 규제가 있는 선형 회귀 실습 - 라쏘

- 훈련 데이터와 테스트 데이터 나누기

```
feature = df[df.keys().drop('charges')].values
outcome = df['charges'].values

from sklearn.model_selection import train_test_split

train_input, test_input, train_target, test_target = train_test_split(feature,
outcome, test_size=0.3, random_state=42)
```

- 라쏘 회귀 학습

```
from sklearn.linear_model import Lasso

lasso = Lasso()
lasso.fit(train_input, train_target)
```

7. 규제가 있는 선형 회귀 실습 - 라쏘

◦ 오차 확인

```
y_pred = lasso.predict(test_input)
import numpy as np
from sklearn.metrics import mean_absolute_error, mean_squared_error

print("MAE :",mean_absolute_error(test_target, y_pred))
print("RMSE :",np.sqrt(mean_squared_error(test_target, y_pred)))
```

```
[실행결과]
MAE : 3847.2450014657793
RMSE : 5808.069908779982
```

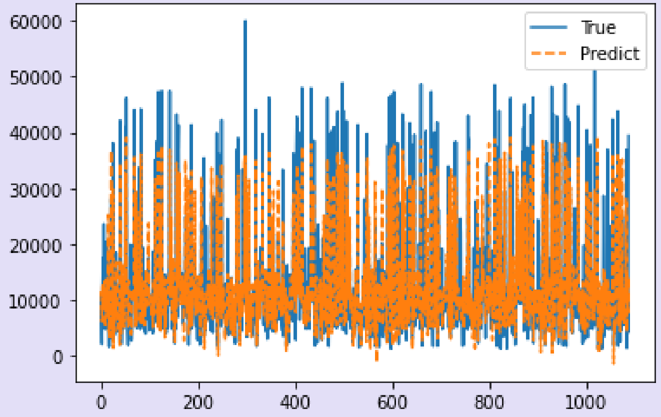
7. 규제가 있는 선형 회귀 실습 - 라쏘

◦ 시각화

```
%matplotlib inline
import matplotlib.pyplot as plt

plt.plot(test_target, linestyle='-',label='True')
plt.plot(y_pred, linestyle='--', label='Predict')
plt.legend()
plt.show()
```

[실행결과]



IMDL
Intelligent Medical Data Lab

7. 규제가 있는 선형 회귀 실습 - 라쏘

- 훈련 데이터와 테스트 데이터 나누기

```
from sklearn.preprocessing import PolynomialFeatures

poly = PolynomialFeatures(degree=2)
feature_poly = poly.fit_transform(feature)

from sklearn.model_selection import train_test_split

train_input, test_input, train_target, test_target = train_test_split(feature_poly,
outcome, test_size=0.3, random_state=42)
```

- 라쏘 회귀 학습

```
from sklearn.linear_model import Lasso

lasso = Lasso()
lasso.fit(train_input, train_target)
```

IMDL Intelligent Medical Data Lab

7. 규제가 있는 선형 회귀 실습 - 라쏘

◦ 오차 확인

```
y_pred = lasso.predict(test_input)
import numpy as np
from sklearn.metrics import mean_absolute_error, mean_squared_error

print("MAE :",mean_absolute_error(test_target, y_pred))
print("RMSE :",np.sqrt(mean_squared_error(test_target, y_pred)))
```

```
[실행결과]
MAE : 2778.638909739709
RMSE : 4731.442182773295
```

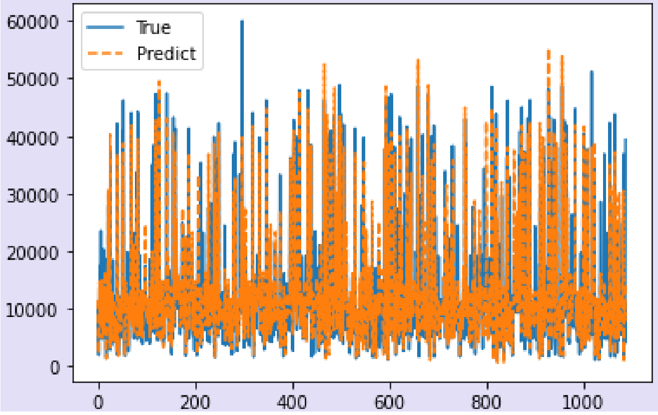
7. 규제가 있는 선형 회귀 실습 - 라쏘

◦ 시각화

```
%matplotlib inline
import matplotlib.pyplot as plt

plt.plot(test_target, linestyle='-',label='True')
plt.plot(y_pred, linestyle='--', label='Predict')
plt.legend()
plt.show()
```

[실행결과]



7. 규제가 있는 선형 회귀 실습 - 라쏘

- 최적의 alpha 구하기

```
from sklearn.model_selection import train_test_split

train_input, test_input, train_target, test_target = train_test_split(feature,
outcome, test_size=0.3, random_state=42)
```

- 라쏘 회귀 학습

```
from sklearn.linear_model import LassoCV

lassocv = LassoCV(alphas=np.arange(0.01,10.01,0.01), cv=5)
lassocv.fit(train_input, train_target)
```

7. 규제가 있는 선형 회귀 실습 - 라쏘

- alpha 확인

```
print(lassocv.alpha_)
```

```
[실행결과]
0.01
```

- 오차 확인

```
y_pred = ridgecv.predict(test_input)
from sklearn.metrics import mean_absolute_error, mean_squared_error

print("MAE :",mean_absolute_error(test_target, y_pred))
print("RMSE :",np.sqrt(mean_squared_error(test_target, y_pred)))
```

```
[실행결과]
MAE : 3846.817930937258
RMSE : 5807.754971810978
```

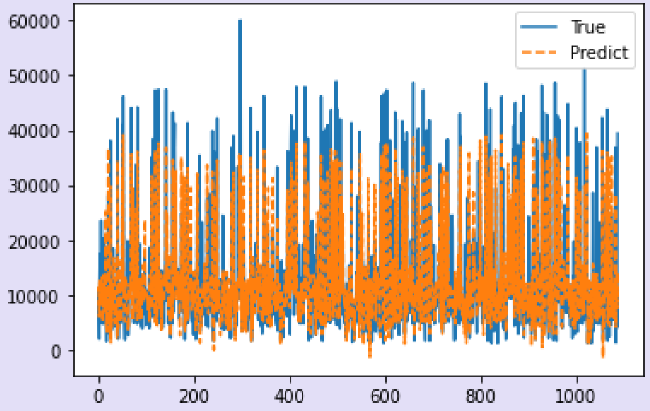
7. 규제가 있는 선형 회귀 실습 - 라쏘

◦ 시각화

```
%matplotlib inline
import matplotlib.pyplot as plt

plt.plot(test_target, linestyle='-',label='True')
plt.plot(y_pred, linestyle='--', label='Predict')
plt.legend()
plt.show()
```

[실행결과]



7. 규제가 있는 선형 회귀 실습 - 엘라스틱넷

◦ 훈련 데이터와 테스트 데이터 나누기

```
feature = df[df.keys().drop('charges')].values
outcome = df['charges'].values

from sklearn.model_selection import train_test_split

train_input, test_input, train_target, test_target = train_test_split(feature, outcome, test_size=0.3, random_state=42)
```

◦ 엘라스틱넷 회귀 학습

```
from sklearn.linear_model import ElasticNet

elasticnet = ElasticNet()
elasticnet.fit(train_input, train_target)
```

7. 규제가 있는 선형 회귀 실습 - 엘라스틱넷

- 오차 확인

```
y_pred = elasticnet.predict(test_input)
import numpy as np
from sklearn.metrics import mean_absolute_error, mean_squared_error

print("MAE :",mean_absolute_error(test_target, y_pred))
print("RMSE :",np.sqrt(mean_squared_error(test_target, y_pred)))
```

```
[실행결과]
MAE : 6553.591880977993
RMSE : 8904.228761924698
```

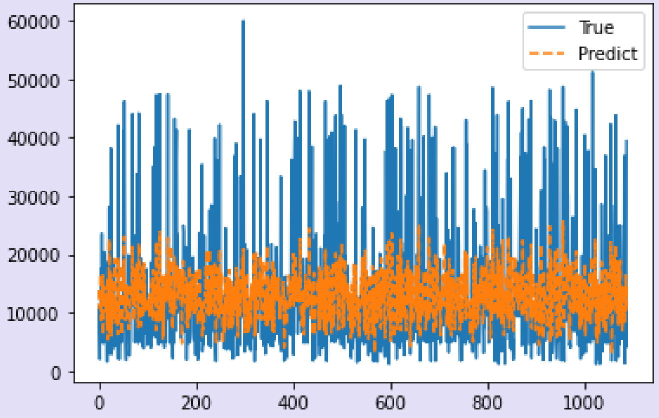
7. 규제가 있는 선형 회귀 실습 - 엘라스틱넷

◦ 시각화

```
%matplotlib inline
import matplotlib.pyplot as plt

plt.plot(test_target, linestyle='-',label='True')
plt.plot(y_pred, linestyle='--', label='Predict')
plt.legend()
plt.show()
```

[실행결과]



IMDL
Intelligent Medical Data Lab

7. 규제가 있는 선형 회귀 실습 - 엘라스틱넷

- 훈련 데이터와 테스트 데이터 나누기

```
from sklearn.preprocessing import PolynomialFeatures

poly = PolynomialFeatures(degree=2)
feature_poly = poly.fit_transform(feature)

from sklearn.model_selection import train_test_split

train_input, test_input, train_target, test_target = train_test_split(feature_poly, outcome, test_size=0.3, random_state=42)
```

- 엘라스틱넷 회귀 학습

```
from sklearn.linear_model import ElasticNet

elasticnet = ElasticNet()
elasticnet.fit(train_input, train_target)
```

7. 규제가 있는 선형 회귀 실습 - 엘라스틱넷

◦ 오차 확인

```
y_pred = elasticnet.predict(test_input)
import numpy as np
from sklearn.metrics import mean_absolute_error, mean_squared_error

print("MAE :",mean_absolute_error(test_target, y_pred))
print("RMSE :",np.sqrt(mean_squared_error(test_target, y_pred)))
```

```
[실행결과]
MAE : 2952.794682978666
RMSE : 4992.398080641136
```

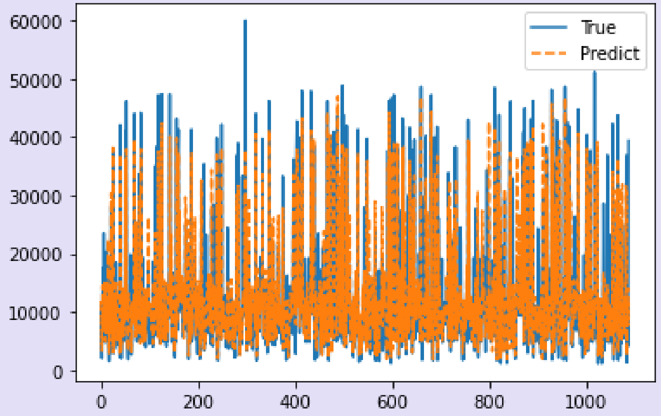
7. 규제가 있는 선형 회귀 실습 - 엘라스틱넷

◦ 시각화

```
%matplotlib inline
import matplotlib.pyplot as plt

plt.plot(test_target, linestyle='-',label='True')
plt.plot(y_pred, linestyle='--', label='Predict')
plt.legend()
plt.show()
```

[실행결과]



7. 규제가 있는 선형 회귀 실습 - 엘라스틱넷

- 최적의 alpha 구하기

```
from sklearn.model_selection import train_test_split

train_input, test_input, train_target, test_target = train_test_split(feature,
outcome, test_size=0.3, random_state=42)
```

- 알레스틱넷 학습

```
from sklearn.linear_model import ElasticNetCV

elasticnetcv = ElasticNetCV(alphas=np.arange(0.01,10.01,0.01), cv=5)
elasticnetcv.fit(train_input, train_target)
```

7. 규제가 있는 선형 회귀 실습 - 라쏘

◦ alpha 확인

```
print(elasticnetcv.alpha_)
```

```
[실행결과]
0.01
```

◦ 오차 확인

```
y_pred = ridgecv.predict(test_input)
from sklearn.metrics import mean_absolute_error, mean_squared_error

print("MAE :",mean_absolute_error(test_target, y_pred))
print("RMSE :",np.sqrt(mean_squared_error(test_target, y_pred)))
```

```
[실행결과]
MAE : 3911.4207944548025
RMSE : 5839.560155249173
```

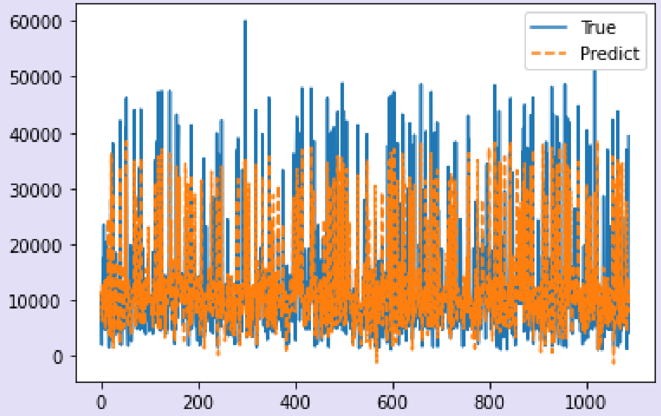
7. 규제가 있는 선형 회귀 실습 - 라쏘

◦ 시각화

```
%matplotlib inline
import matplotlib.pyplot as plt

plt.plot(test_target, linestyle='-',label='True')
plt.plot(y_pred, linestyle='--', label='Predict')
plt.legend()
plt.show()
```

[실행결과]



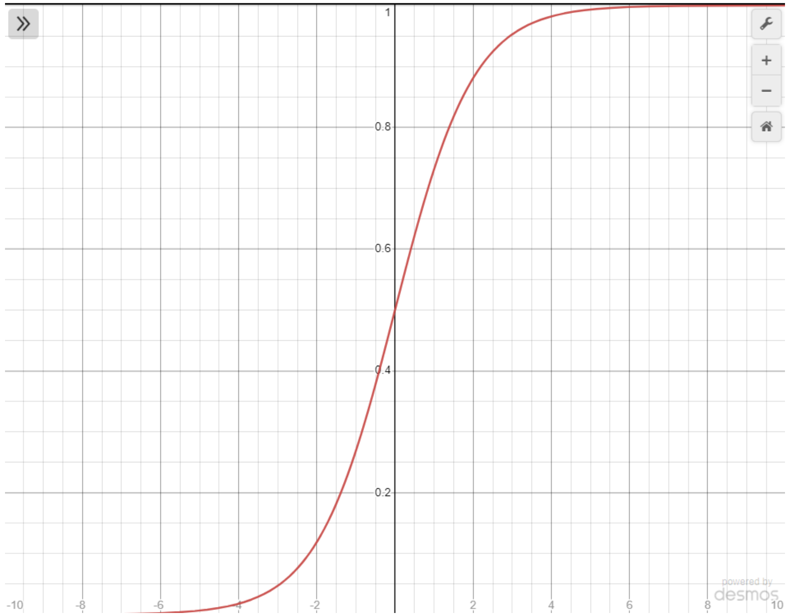
IMDL
Intelligent Medical Data Lab

8. 로지스틱 회귀

- 로지스틱 회귀
  - 이진 분류를 하기 위해 logit 변환을 통해 0과 1로 분류

$$y = \sigma(\theta^T X)$$

$$\sigma(x) = \frac{1}{1 + \exp^{-x}}$$



https://www.desmos.com/calculator/diikd8egvj

IMDL Intelligent Medical Data Lab

9. 로지스틱 회귀 실습

◦ 데이터셋 로드

```
import pandas as pd

df = pd.read_csv('fish.csv')
df.head()
```

[실행결과]

	Species	Weight	Length
0	Bream	242.0	25.4
1	Bream	290.0	26.3
2	Bream	340.0	26.5
3	Bream	363.0	29.0
4	Bream	430.0	29.0

IMDL Intelligent Medical Data Lab

9. 로지스틱 회귀 실습

◦ 데이터 전처리

```
feature = df[['Weight','Length']].values
df.loc[df['Species']=='Bream','Species']=0
df.loc[df['Species']=='Smelt','Species']=1
df['Species'] = df['Species'].astype('int32')
outcome = df['Species'].values
```

◦ 훈련 데이터와 테스트 데이터 나누기

```
from sklearn.model_selection import train_test_split

train_input, test_input, train_target, test_target = train_test_split(feature,
outcome, test_size=0.3, random_state=42)
```

9. 로지스틱 회귀 실습

◦ 로지스틱 회귀 학습

```
from sklearn.linear_model import LogisticRegression

lr = LogisticRegression()
lr.fit(train_input, train_target)
```

◦ 성능 측정

```
y_pred = lr.predict(test_input)
from sklearn.metrics import accuracy_score

print("Accuracy :",accuracy_score(test_target,y_pred))
```

```
[실행결과]
Accuracy : 1.0
```

9. 로지스틱 회귀 실습

- 훈련 데이터와 테스트 데이터 나누기

```
from sklearn.model_selection import train_test_split

train_input, test_input, train_target, test_target = train_test_split(feature,
outcome, test_size=0.3, random_state=42)
```

- 데이터 스케일 수행

```
from sklearn.preprocessing import MinMaxScaler

feature_scaler = MinMaxScaler()
train_input_scaled = feature_scaler.fit_transform(train_input)
test_input_scaled = feature_scaler.transform(test_input)
```

9. 로지스틱 회귀 실습

◦ 로지스틱 회귀 학습

```
from sklearn.linear_model import LogisticRegression

lr = LogisticRegression()
lr.fit(train_input_scaled, train_target)
```

◦ 성능 측정

```
y_pred = lr.predict(test_input_scaled)
from sklearn.metrics import accuracy_score

print("Accuracy :",accuracy_score(test_target,y_pred))
```

```
[실행결과]
Accuracy : 1.0
```

9. 로지스틱 회귀 실습

- 특성 추가

```
from sklearn.preprocessing import PolynomialFeatures

poly = PolynomialFeatures(degree=2)
feature_poly = poly.fit_transform(feature)
```

- 훈련 데이터와 테스트 데이터 나누기

```
from sklearn.model_selection import train_test_split

train_input, test_input, train_target, test_target = train_test_split(feature_poly,
outcome, test_size=0.3, random_state=42)
```

9. 로지스틱 회귀 실습

- 데이터 스케일 수행

```
from sklearn.preprocessing import MinMaxScaler

feature_scaler = MinMaxScaler()
train_input_scaled = feature_scaler.fit_transform(train_input)
test_input_scaled = feature_scaler.transform(test_input)
```

- 로지스틱 회귀 학습

```
from sklearn.linear_model import LogisticRegression

lr = LogisticRegression()
lr.fit(train_input_scaled , train_target)
```

9. 로지스틱 회귀 실습

◦ 성능 측정

```
y_pred = lr.predict(test_input_scaled)
from sklearn.metrics import accuracy_score

print("Accuracy :",accuracy_score(test_target,y_pred))
```

```
[실행결과]
Accuracy : 1.0
```

IMDL Intelligent Medical Data Lab

10. Homework

◦ 스스로 해보기

- Medical Insurance Premium Prediction 데이터셋을 이용하여 선형 회귀를 이용하여 예측하여라. 선형회귀, 릿지 회귀, 라쏘 회귀, 엘라스틱 회귀 중 한 가지 알고리즘 사용

	A	B	C	D	E	F	G	H	I	J	K	L
1	Age	Diabetes	BloodPres	AnyTransp	AnyChron	Height	Weight	KnownAlle	HistoryOf(	NumberOf	PremiumPrice	
2	45	0	0	0	0	155	57	0	0	0	25000	
3	60	1	0	0	0	180	73	0	0	0	29000	
4	36	1	1	0	0	158	59	0	0	1	23000	
5	52	1	1	0	1	183	93	0	0	2	28000	
6	38	0	0	0	1	166	88	0	0	1	23000	
7	30	0	0	0	0	160	69	1	0	1	23000	
8	33	0	0	0	0	150	54	0	0	0	21000	
9	23	0	0	0	0	181	79	1	0	0	15000	
10	48	1	0	0	0	169	74	1	0	0	23000	

10. Homework

- 스스로 해보기
  - Pima Indians Diabetes Database 데이터셋을 이용하여 로지스틱 회귀를 이용하여 이진 분류를 수행하여라.

	A	B	C	D	E	F	G	H	I
1	Pregnanci	Glucose	BloodPres	SkinThickr	Insulin	BMI	DiabetesP	Age	Outcome
2	6	148	72	35	0	33.6	0.627	50	1
3	1	85	66	29	0	26.6	0.351	31	0
4	8	183	64	0	0	23.3	0.672	32	1
5	1	89	66	23	94	28.1	0.167	21	0
6	0	137	40	35	168	43.1	2.288	33	1
7	5	116	74	0	0	25.6	0.201	30	0
8	3	78	50	32	88	31	0.248	26	1
9	10	115	0	0	0	35.3	0.134	29	0
10	2	197	70	45	543	30.5	0.158	53	1