1. 회귀분석 실습

A. 캘리포니아 주택 가격 데이터 분석

```
import pandas as pd

df = pd.read_csv('d:/data/house/housing.csv')
df.head()
```

```
#탐색적 데이터 분석(Exploratory data analysis, EDA)
```

```
#바다 근접 특성
df["ocean_proximity"].value_counts()
```

```
#히스토그램
%matplotlib inline
import matplotlib.pyplot as plt
df.hist(bins=50, figsize=(20,15))
plt.show()
```

```
#랜덤시드 고정
import numpy as np
np.random.seed(10)
```

```
#지리적 데이터 시각화
ax = df.plot(kind="scatter", x="longitude", y="latitude")
ax.set(xlabel='longitude', ylabel='latitude') #경도, 위도
```

```
#데이터 포인트가 밀집된 지역을 표현하기 위하여 투명도 조절
ax = df.plot(kind="scatter", x="longitude", y="latitude",
alpha=0.1)
ax.set(xlabel='longitude', ylabel='latitude') #경도, 위도
```

```
#s 포인트 사이즈:인구, c: 주택가격, jet: blue~red 컬러 옵션, sh
arex=False x축 라벨을 공유하지 않음
ax = df.plot(kind="scatter", x="longitude", y="latitude", a
lpha=0.4, s=df["population"]/100, label="population", figsi
ze=(10,7), c="median_house_value", cmap=plt.get_cmap("jet
"), colorbar=True, sharex=False)
ax.set(xlabel='longitude', ylabel='latitude') #경도, 위도
plt.legend()
```

```
#산점도(x: 중위소득, y:중위주택가격)
df.plot(kind="scatter", x="median_income", y="median_house_
value", alpha=0.1)
```

```
df_cat = df['ocean_proximity']
df_cat.head(10)
```

```
#원핫인코딩
df2 = pd.get_dummies(data=df, columns=['ocean_proximity'])
df2
```

```
#pip install missingno
%matplotlib inline
import missingno as msno
msno.matrix(df)
#흰색 - 결측값
#스파크라인(spark line) - 각 샘플의 데이터 완성도를 표현
msno.bar(df) #필드별로 결측값이 얼마나 있는지 확인
```

```
#결측값을 각각의 변수의 중위수로 치환
from sklearn.impute import SimpleImputer

imputer = SimpleImputer(strategy="median")

imputer.fit(df2)
X = imputer.transform(df2)
df_tr = pd.DataFrame(X, columns=df2.columns)
df_tr.tail()
```

```
#평균주택가격을 y로 사용
# 레이블로 사용
housing_labels = df_tr["median_house_value"].copy()
# 레이블 삭제
df_tr.drop("median_house_value", axis=1, inplace = True)
```

```
#정규화
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
scaler.fit(df_tr)
scaled_df = scaler.transform(df_tr)
scaled_df.shape
```

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(scaled_
df, housing_labels, test_size=0.2, random_state=0)
print(X_train.shape, X_test.shape)
```

```
from sklearn.linear_model import LinearRegression

lin_reg = LinearRegression()
lin_reg.fit(X_train, y_train)
```

```
from sklearn.metrics import mean_squared_error
y_pred = lin_reg.predict(X_test)
# 평균제곱근오차 : 예측값과 실제값을 뺀 후 제곱한 값들을 다 더하
고 n으로 나눈 값의 제곱근(평균제곱오차의 제곱근)
rms = np.sqrt(mean_squared_error(y_test, y_pred))
print(rms)
#오차가 너무 크므로 다른 모형 사용
```

```
from sklearn.tree import DecisionTreeRegressor

tree_reg = DecisionTreeRegressor()
tree_reg.fit(X_train, y_train)

y_pred = tree_reg.predict(X_test)
rms = np.sqrt(mean_squared_error(y_test, y_pred))
print(rms)
```

```
from sklearn.svm import SVR
#Support Vector Regression
svm_reg = SVR(kernel="linear")
svm_reg.fit(X_train, y_train)

y_pred = svm_reg.predict(X_test)
rms = np.sqrt(mean_squared_error(y_test, y_pred))
print(rms)
```

```
from sklearn.ensemble import RandomForestRegressor
forest_reg = RandomForestRegressor(n_jobs = -1)
forest_reg.fit(X_train, y_train)

y_pred = forest_reg.predict(X_test)
rms = np.sqrt(mean_squared_error(y_test, y_pred))
print(rms)
#가장 좋은 성능
```

```
#파라미터 튜닝(시간이 오래 걸리는 작업)
from sklearn.model_selection import GridSearchCV
#max_features : 트리를 만드는데 사용할 변수의 갯수
param_grid = [
    {'n_estimators': [3, 10, 30, 50, 100], 'max_features':
[2, 4, 6, 8]},
  ]
# n_jobs=-1 모든 cpu core 사용
forest_reg = RandomForestRegressor(random_state=0,
n_jobs=-1)
grid_search = GridSearchCV(forest_reg, param_grid, cv=5)
grid_search.fit(X_train, y_train)
```

```
#실험을 통해 얻은 최적의 파라미터
grid_search.best_params_
```

```
#최적의 모형
grid_search.best_estimator_
```

```
#랜덤 포레스트 모형을 만드는데 가장 기여도가 큰 변수 확인
feature_importances = grid_search.best_estimator_.feature_i
mportances_
plt.bar(range(len(feature_importances)), feature_importance
s)
#인덱스 7(8번째 변수) 평균수입(median_income)
```

```
#RandomizedSearchCV() 최근에 많이 사용되고 있는 파라미터 최적화
함수
#매개변수의 범위를 지정해주면 무작위로 매개변수를 조합하여 최적의
성능 측정
from sklearn.model_selection import RandomizedSearchCV
from scipy.stats import randint

param_distribs = {
        'n_estimators': randint(low=1, high=100),
        'max_features': randint(low=1, high=8),
    }

forest_reg = RandomForestRegressor(random_state=0, n_jobs=-
1)
rnd_search = RandomizedSearchCV(forest_reg, param_distribut
ions=param_distribs, cv=5, random_state=0)
rnd_search.fit(X_train, y_train)
```

```
cvres = rnd_search.cv_results_
for mean_score, params in zip(cvres["mean_test_score"],
cvres["params"]):
    print(np.sqrt(mean_score), params)
```

```
rnd_search.best_estimator_
```

```
final_model = rnd_search.best_estimator_
y_pred = final_model.predict(X_test)
rms = np.sqrt(mean_squared_error(y_test, y_pred))
print(rms)
```

```
#결론 : 주택가격에 가장 큰 영향을 주는 특성은 평균소득, 예측오차는 약 48929달러
```

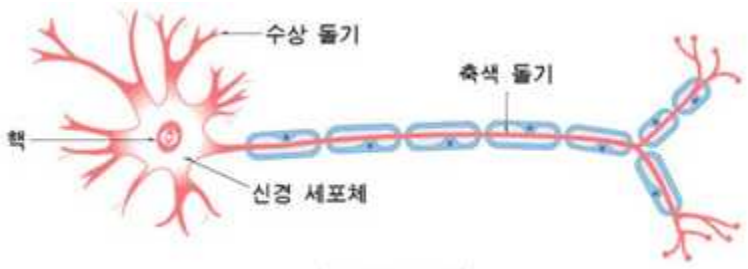
2. 인공신경망

A. 인공신경망의 개요

1) 개요

① 인공지능의 한 분야

② 인간 두뇌의 생물학적 작동 형태를 모방하여 컴퓨터로 하여금 지적인 능력을 갖추게 하는 방법론



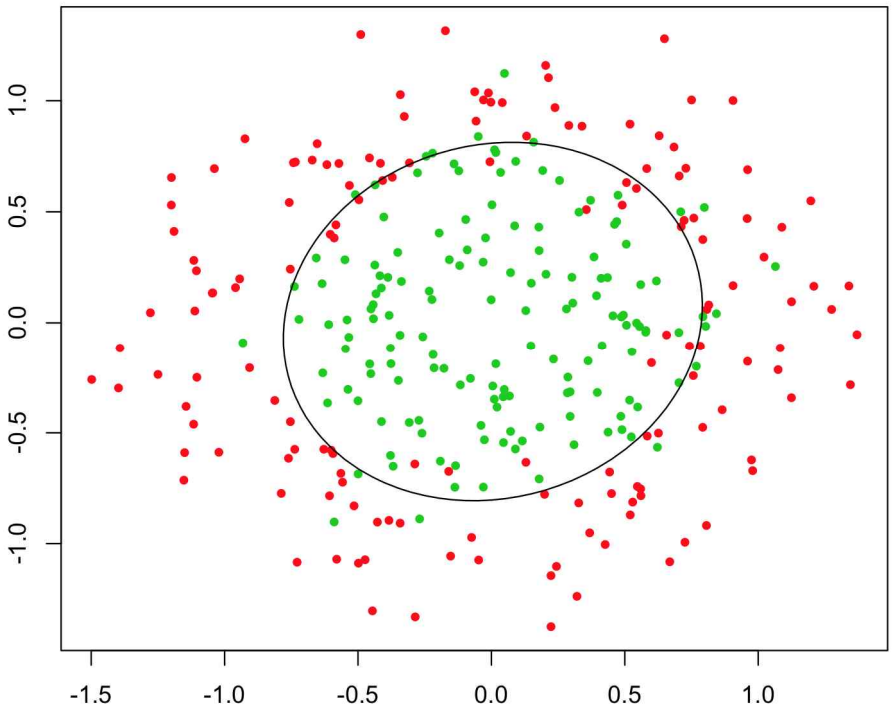
뉴런 : 화학적 또는 전기적 신호를 전송하고 처리하는 세포, 다른 뉴런과 연결되어 네트워크를 만든다.

모든 뉴런에는 입력(수상돌기), 세포체, 출력(축색돌기)이 있다.

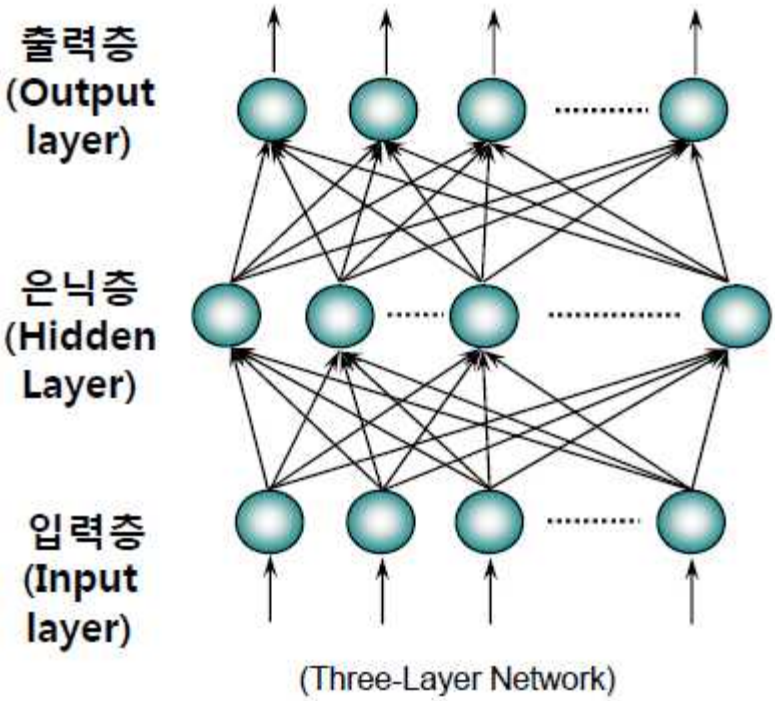
뉴런이 가질 수 있는 입력은 10,000개로 인공신경망에 비해 훨씬 더 복잡하다.

③ 공학 부문에서 시작되어 재무 관리 등에 도입되기 시작

2) 선형분류와 비선형분류

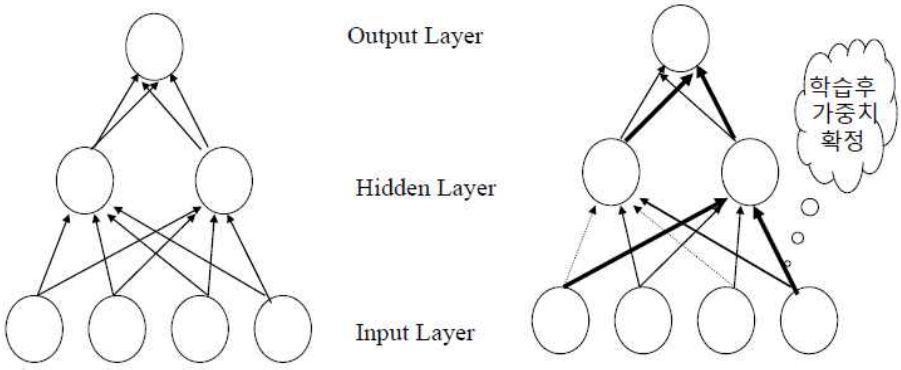


3) 인공신경망의 구조



(Three-Layer Network)

4) 학습결과의 예



Before Training

After Training

B. 인공신경망의 기본 개념

1) 노드(유닛)

① 각각의 인자(neuron)를 의미

2) 층(layer)

① 보통 세 개의 층(three-layered)

② 은닉(hidden) 층

입력값과 출력값을 연결시켜 주는 매개변수값으로 채워지게 되는데 전체 신경망의 성공여부가 바로 이 은닉층의 역할에 달려 있다고 할 수 있음

3) 가중치(weight)

① i번째 노드와 j번째 노드와의 결합정도

② Wij: 층과 층 사이의 연결이 i에서 j로 갈 때의 연결강도

C. 인공신경망 추론과정

1) 1단계: 초기 연결가중치 결정

연결가중치를 임의의 아주 작은 값(보통 -1~1 사이)으로 초기화

2) 2단계: 전방향 계산

① 은닉층 및 출력층에서 입력값에 연결가중치를 곱하여 각 처리요소들의 출력값을 계산

② 전이함수를 사용하여 출력값 결정

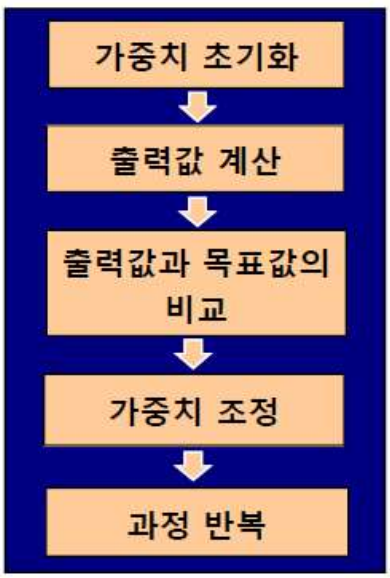
3) 3단계: 역방향 계산

① 출력층의 출력값과 목표출력값 사이의 오류치 계산

② 출력층과 은닉층 사이의 연결 가중치를 수정

③ 은닉층과 입력층 사이의 연결 가중치를 수정

4) 4단계: Epoch (2,3 단계)의 반복



3. 인공신경망(회귀분석) 실습

A. 단층퍼셉트론

```
%matplotlib inline

import numpy as np
import matplotlib.pyplot as plt

# -1 ~ 1 까지 0.2 간격으로 10개
x = np.arange(-1.0, 1.0, 0.2)
y = np.arange(-1.0, 1.0, 0.2)

# 출력값을 저장할 10x10 그리드
z = np.zeros((10,10))

# x, y 값의 가중치(가중치값을 바꿔가면서 테스트)
w_x = 2.5
w_y = 3.0

# w_x = 0
# w_y = 3.0

# w_x = 2.5
# w_y = 0

# 편향(편향값을 바꿔가면서 테스트)
#bias = -2
bias = 0.1
#bias = 2
```

```
# 그리드맵의 각 그리드별 뉴런의 연산
for i in range(10):
    for j in range(10):

        # x에 가중치를 곱하고 편향을 더한 값
        u = x[i]*w_x + y[j]*w_y + bias

        # 활성화 함수(시그모이드 함수)
        y0 = 1/(1+np.exp(-u))
        # 출력값 저장
        z[j][i] = y0

# 그리드맵 표시
plt.imshow(z, "gray", vmin = 0.0, vmax = 1.0)
plt.colorbar()
plt.show()
```

B. 단층신경망

```
from sklearn.datasets import load_diabetes
diabetes = load_diabetes()
##############################
print(diabetes.data.shape, diabetes.target.shape)
##############################
diabetes.data[0:3]
##############################
diabetes.target[:3] #당수치
##############################
#bmi와 당수치는 양의 상관관계
import matplotlib.pyplot as plt
plt.scatter(diabetes.data[:, 2], diabetes.target)
plt.xlabel('x') #bmi
plt.ylabel('y') #당수치
plt.show()
##############################
X = diabetes.data[:, 2]
y = diabetes.target
##############################
```

```
class Model:
    def __init__(self):
        self.w = 1.0     # 가중치 초기화
        self.b = 1.0     # 편향 초기화

    def forward(self, x):
        y_hat = x * self.w + self.b
        return y_hat

    def backward(self, x, err):
        w_grad = x * err
        b_grad = 1 * err
        return w_grad, b_grad

    def fit(self, x, y, epochs=100):
        for i in range(epochs):           # epoch만큼 반복
            for x_i, y_i in zip(x, y):    # 모든 샘플 반복
                y_hat = self.forward(x_i) # 순전파
                # 오차 계산(실제값 - 예측값)
                err = -(y_i - y_hat)
                # 역전파
                w_grad, b_grad = self.backward(x_i, err)
                self.w -= w_grad          # 가중치 수정
                self.b -= b_grad          # 절편 수정
##############################
model = Model()
model.fit(X, y)
##############################
```

```
%matplotlib inline
import matplotlib.pyplot as plt
plt.scatter(X, y)
pt1 = (-0.1, -0.1 * model.w + model.b)
pt2 = (0.15, 0.15 * model.w + model.b)
plt.plot([pt1[0], pt2[0]], [pt1[1], pt2[1]], c='r') #회귀선
plt.xlabel('X')
plt.ylabel('y')
plt.show()
```

C. 회귀분석

```
%matplotlib inline

import numpy as np
import matplotlib.pyplot as plt

# -1 ~ 1 0.2 간격의 10개의 리스트
x = np.arange(-1.0, 1.0, 0.2)
y = np.arange(-1.0, 1.0, 0.2)

# 출력을 저장하는 10X10 그리드
z = np.zeros((10,10))

# 가중치
w_im = np.array([[4.0,4.0],
                 [4.0,4.0]])  #hidden layer에 사용할 가중치 행렬
w_mo = np.array([[1.0],
                 [-1.0]])     #output layer에 사용할 가중치 행렬

# 편향
b_im = np.array([3.0,-3.0]) # 은닉층
b_mo = np.array([0.1])      # 출력층

# 은닉층
def hidden_layer(x, w, b):
    u = np.dot(x, w) + b # 행렬곱셈
    return 1/(1+np.exp(-u)) # 활성화함수(시그모이드 함수)
```

```
# 출력층
def output_layer(x, w, b):
    u = np.dot(x, w) + b
    return u  # 활성화함수(항등함수)

# 그리드맵의 각 그리드별 신경망 연산
for i in range(10):
    for j in range(10):

        # 순전파
        inp = np.array([x[i], y[j]])        # 입력층
        hid = hidden_layer(inp, w_im, b_im) # 은닉층
        out = output_layer(hid, w_mo, b_mo) # 출력층

        # 그리드맵에 신경망 출력 값 저장
        z[j][i] = out[0]

print(z[:2])
# 그리드맵으로 표시
plt.imshow(z, "gray", vmin = 0.0, vmax = 1.0)
plt.colorbar()
plt.show()
```

```
#인위적으로 만든 나이와 키 데이터로 나이와 키의 상관관계를
분석하는 회귀분석 실습
import numpy as np

# 데이터 생성
np.random.seed(seed=1) # 난수 고정
X_min = 4 # X의 하한
X_max = 30 # X의 상한
X_n = 16

# 랜덤으로 만든 나이 데이터
X = 5 + 25 * np.random.rand(X_n)
Prm_c = [170, 108, 0.2] # 생성 매개 변수
# 랜덤으로 만든 신장 데이터
T = Prm_c[0] - Prm_c[1] * np.exp(-Prm_c[2] * X) + 4 * np.ran
dom.randn(X_n)
============================
%matplotlib inline
import matplotlib.pyplot as plt

plt.figure(figsize=(4, 4))
plt.scatter(X, T, marker='o')
plt.xlim(X_min, X_max)
plt.grid(True)
plt.show()
============================
```

```
from mpl_toolkits.mplot3d import Axes3D

# 평균제곱오차 함수
# x 나이, t 키, w 기울기와 편차
def mse(x, t, w):
    # 예측값: x에 기울기를 곱하고 편차를 더한 값
    y = w[0] * x + w[1]
    return np.mean((y - t)**2)

xn = 100 # 등고선 표시 해상도
w0_range = [-25, 25]
w1_range = [120, 170]
x0 = np.linspace(w0_range[0], w0_range[1], xn)
x1 = np.linspace(w1_range[0], w1_range[1], xn)
# 직사각형 격자를 만들고
xx0, xx1 = np.meshgrid(x0, x1)
J = np.zeros((len(x0), len(x1)))
for i0 in range(xn):
    for i1 in range(xn):
        #평균제곱오차를 계산하여 리스트에 저장
        J[i1, i0] = mse(X, T, (x0[i0], x1[i1]))

plt.figure(figsize=(9, 4))
#3d 차트
ax = plt.subplot(1, 2, 1, projection='3d') #1행 2열 중 첫번째
서브플롯
# rstride : row stride 행방향 이동, cstride : column stride
열방향 이동
# 곡면플롯
ax.plot_surface(xx0, xx1, J, rstride=10, cstride=10, alpha=
0.3,color='blue', edgecolor='black')
```

```
ax.set_xticks([-20, 0, 20])
ax.set_yticks([120, 140, 160])
ax.view_init(20, -60)
plt.subplot(1, 2, 2) #1행 2열 중 두번째 서브플롯
#등고선
cont = plt.contour(xx0, xx1, J, 30, colors='black',
                   levels=[100, 1000, 10000, 100000])
#등고선 플롯의 고도각(Elevation)에 레이블 지정
cont.clabel(fmt='%d', fontsize=8)
plt.grid(True)
plt.show()

# 왼쪽 그래프 : 평균 제곱 오차는 계곡 모양, 기울기의 변화에 따라 평균제곱오차가 크게 바뀜
# 오른쪽 그래프 : 기울기의 변화에 따라 등고선의 간격으로 오차를 확인할 수 있음, 계곡의 최저점은 135 근처
============================
from sklearn.linear_model import LinearRegression
lin_reg = LinearRegression()
lin_reg.fit(X.reshape(-1,1), T)
============================
print("예측:", lin_reg.predict(X.reshape(-1,1)))
print('실제:', T)
============================
```

```
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error

pred = lin_reg.predict(X.reshape(-1,1))
#평균제곱오차
lin_mse = mean_squared_error(T, pred)
print(lin_mse)

#평균제곱근오차
lin_rmse = np.sqrt(lin_mse)
print(lin_rmse)

#평균절대오차
lin_mae = mean_absolute_error(T, pred)
print(lin_mae)
============================
```

```
from scipy import stats
slope,intercept,r_value,p_value,stderr=stats.linregress(X,
T)
slope,intercept,r_value,p_value,stderr

# 회귀선 표시
def show_line(w):
    xb = np.linspace(X_min, X_max, 100)
    y = w[0] * xb + w[1]
    plt.plot(xb, y, color=(.5, .5, .5), linewidth=4)

plt.figure(figsize=(4, 4))
W=np.array([slope, intercept])
show_line(W)
plt.scatter(X, T, marker='o')
plt.xlim(X_min, X_max)
plt.grid(True)
plt.show()
============================
```

```
#독립변수가 1개 아닌 2개 이상인 경우(다중회귀분석)
# 독립변수: 나이, 체중
# 종속변수: 키
X0 = X
X0_min = 5
X0_max = 30
np.random.seed(seed=1) # 난수를 고정
#랜덤으로 만든 몸무게 데이터
X1 = 23 * (T / 100)**2 + 2 * np.random.randn(X_n)
X1_min = 40
X1_max = 75
============================
print(np.round(X0, 2)) #나이
print(np.round(X1, 2)) #체중
print(np.round(T, 2)) #키
============================
#나이,체중,키 3차원 그래프
def show_data2(ax, x0, x1, t):
    for i in range(len(x0)):
        ax.plot([x0[i], x0[i]], [x1[i], x1[i]],
                [120, t[i]], color='gray')
        ax.plot(x0, x1, t, 'o',
                markersize=6, markeredgewidth=0.5)
        ax.view_init(elev=35, azim=-75)


plt.figure(figsize=(6, 5))
ax = plt.subplot(1,1,1,projection='3d')
show_data2(ax, X0, X1, T)
plt.show()
============================
```

```
#면의 표시
def show_plane(ax, w):
    px0 = np.linspace(X0_min, X0_max, 5)
    px1 = np.linspace(X1_min, X1_max, 5)
    px0, px1 = np.meshgrid(px0, px1)
    y = w[0]*px0 + w[1] * px1 + w[2]
    ax.plot_surface(px0, px1, y, rstride=1, cstride=1, alpha
=0.3,color='blue', edgecolor='black')

#면의 MSE
def mse_plane(x0, x1, t, w):
    y = w[0] * x0 + w[1] * x1 + w[2]
    mse = np.mean((y - t)**2)
    return mse

plt.figure(figsize=(6, 5))
ax = plt.subplot(1, 1, 1, projection='3d')
W = [1.5, 1, 90]
show_plane(ax, W)
show_data2(ax, X0, X1, T)
mse = mse_plane(X0, X1, T, W)
plt.show()

print('mse:',mse) # 165.7
============================
```

```
lin_reg = LinearRegression()
lin_reg.fit(list(zip(X0,X1)), T)
pred = lin_reg.predict(list(zip(X0,X1)))
#평균제곱오차
lin_mse = mean_squared_error(T, pred)
print(lin_mse)

#평균제곱근오차
lin_rmse = np.sqrt(lin_mse)
print(lin_rmse)

#평균절대오차
lin_mae = mean_absolute_error(T, pred)
print(lin_mae)
```

D. 오존량 예측

```
import pandas as pd
df=pd.read_csv('c:/data/ozone/ozone2.csv')
df.head()
====================================
X=df[['Solar.R','Wind','Temp']]
y=df['Ozone']
====================================
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_siz
e=0.2,random_state=7)
====================================
from keras.models import Sequential
from keras.layers import Dense
model=Sequential()
model.add(Dense(8,input_shape=(len(X_train.columns),),activ
ation='relu'))
model.add(Dense(4,activation='relu'))
model.add(Dense(2,activation='relu'))
model.add(Dense(1))
model.compile(loss='mse', optimizer='adam',
metrics=['mse','mae'])
model.summary()
====================================
```

```
X_train.boxplot()
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
scaler.fit(X_train)
X_train_scaled=scaler.transform(X_train)
X_test_scaled=scaler.transform(X_test)
X_train_scaled=pd.DataFrame(X_train_scaled)
X_test_scaled=pd.DataFrame(X_test_scaled)
====================================
import matplotlib.pyplot as plt
X_train_scaled.boxplot()
plt.show()
====================================
from keras.callbacks import EarlyStopping
early_stopping = EarlyStopping(monitor='val_loss',
patience=50)
hist=model.fit(X_train_scaled,y_train,epochs=500,validation
_split=0.2,callbacks=[early_stopping])
====================================
plt.plot(hist.history['loss'])
plt.plot(hist.history['val_loss'])
plt.legend(['loss','val_loss'])
plt.show()
====================================
plt.plot(hist.history['mse'])
plt.plot(hist.history['val_mse'])
plt.legend(['mse','val_mse'])
plt.show()
====================================
```

```
plt.plot(hist.history['mae'])
plt.plot(hist.history['val_mae'])
plt.legend(['mae','val_mae'])
plt.show()
==================================
scores=model.evaluate(X_train_scaled,y_train)
print(scores)

scores=model.evaluate(X_test_scaled,y_test)
print(scores)
==================================
from sklearn.metrics import r2_score
pred=model.predict(X_test_scaled)
r2_score(y_test, pred)
```

E. 당뇨병 예측

```
from sklearn.datasets import load_diabetes
diabetes=load_diabetes()
==================================
X=diabetes.data
y=diabetes.target
==================================
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_siz
e=0.2,random_state=7)
==================================
from keras.models import Sequential
from keras.layers import Dense
model=Sequential()
model.add(Dense(24,input_shape=(X_train.shape[1],),activati
on='relu'))
model.add(Dense(12,activation='relu'))
model.add(Dense(6,activation='relu'))
model.add(Dense(1))
model.compile(loss='mse', optimizer='adam', metrics=['mse',
'mae'])
model.summary()
==================================
import pandas as pd
pd.DataFrame(X_train).boxplot()
==================================
```

```
from keras.callbacks import EarlyStopping
early_stopping = EarlyStopping(monitor='val_loss', patience
=50)
hist=model.fit(X_train,y_train,epochs=500,validation_split=
0.2,callbacks=[early_stopping])
==================================
import matplotlib.pyplot as plt
plt.plot(hist.history['loss'])
plt.plot(hist.history['val_loss'])
plt.legend(['loss','val_loss'])
plt.show()
==================================
plt.plot(hist.history['mse'])
plt.plot(hist.history['val_mse'])
plt.legend(['mse','val_mse'])
plt.show()
==================================
plt.plot(hist.history['mae'])
plt.plot(hist.history['val_mae'])
plt.legend(['mae','val_mae'])
plt.show()
==================================
scores=model.evaluate(X_train,y_train)
print(scores)
==================================
scores=model.evaluate(X_test,y_test)
print(scores)
==================================
from sklearn.metrics import r2_score
pred=model.predict(X_test)
r2_score(y_test, pred)
```