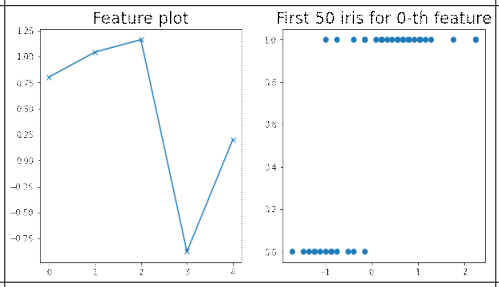
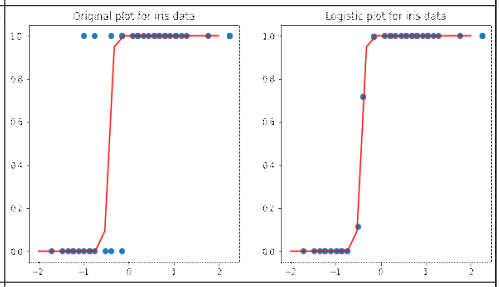
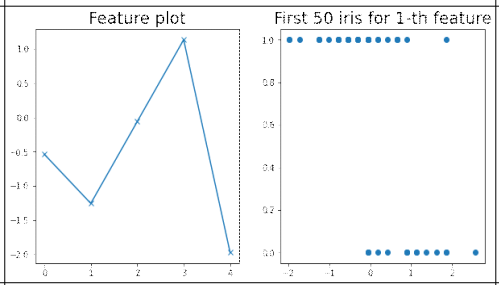
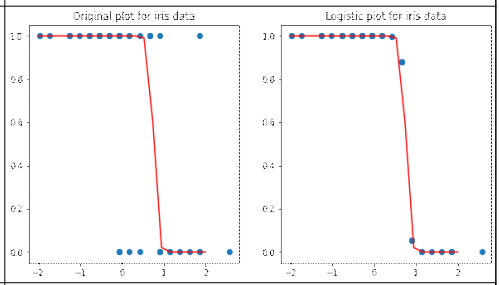
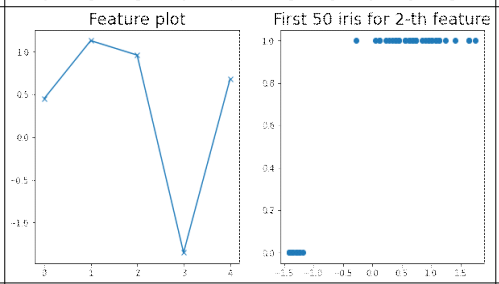
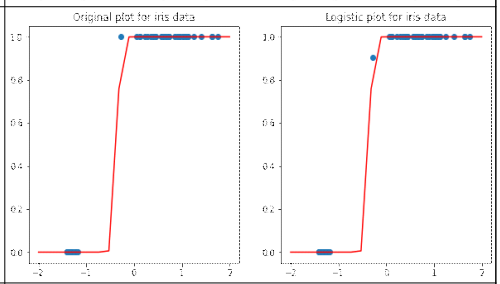
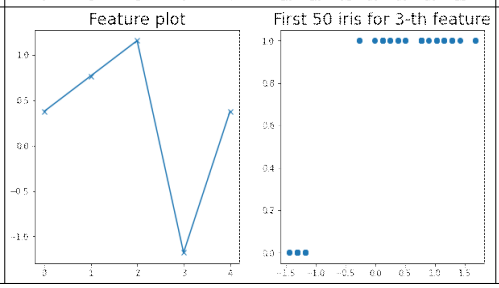
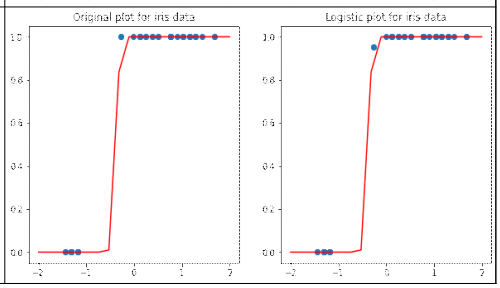
기계학습 중간고사

2020020490
고려대학교 기계공학과 석사과정 문용환
moonyh1230@kist.re.kr

1. 붓꽃 데이터를 이용한 logistic regression

sklearn data 패키지의 iris_data를 활용하여 붓꽃을 분류하였다. 붓꽃 데이터는 setosa, versicolor, virginica의 세 가지 붓꽃 종(species)으로 구성된 target data와 꽃받침 길이와 폭, 꽃잎 길이와 폭의 4가지로 구성된 feature data로 이루어져 있다. 이 4종류의 feature data를 logistic regression을 통해 분류하고, 결과로 setosa와 그 이외의 종으로 구분하였다.

- Feature data와 target data의 scatter plot 및 logistic regression plot

features	feature - scatter plot	logistic regression plot
Sepal length		
Sepal width		
Petal Length		
Petal width		

- Logistic regression(L2 constraint application)

150개의 data set 중 임의로 선택된 112개의 training data로 나머지 38개의 target을 예측해본 결과는 다음과 같다.

	precision	recall	f1-score	support
0	1.00	1.00	1.00	13
1	1.00	1.00	1.00	25
accuracy			1.00	38
macro avg	1.00	1.00	1.00	38
weighted avg	1.00	1.00	1.00	38

2. 당뇨병 진행도 데이터를 이용한 OLS regression

sklearn data 패키지의 diabetes_data를 활용하여 OLS 회귀분석을 진행하였다. 당뇨병 진행도 데이터는 당뇨병의 진행도를 정수로 나타낸 target data와 나이, 성별, BMI 지수, 평균 혈압, 그리고 6종류의 혈액검사 정보로 구성된 feature data로 이루어져 있다. 주어진 feature data를 OLS 회귀분석을 통해 분석하고, 회귀곡선을 나타내는 공식을 도출하였다.

- OLS regression(result summary)

OLS Regression Results

Dep. Variable:	DP	R-squared:	0.518
Model:	OLS	Adj. R-squared:	0.507
Method:	Least Squares	F-statistic:	46.27
Date:	Thu, 17 Jun 2021	Prob (F-statistic):	3.83e-62
Time:	09:30:01	Log-Likelihood:	-2386.0
No. Observations:	442	AIC:	4794.
Df Residuals:	431	BIC:	4839.
Df Model:	10		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	152.1335	2.576	59.061	0.000	147.071	157.196
age	-10.0122	59.749	-0.168	0.867	-127.448	107.424
sex	-239.8191	61.222	-3.917	0.000	-360.151	-119.488
bmi	519.8398	66.534	7.813	0.000	389.069	650.610
bp	324.3904	65.422	4.958	0.000	195.805	452.976
s1	-792.1842	416.684	-1.901	0.058	-1611.169	26.801
s2	476.7458	339.035	1.406	0.160	-189.621	1143.113
s3	101.0446	212.533	0.475	0.635	-316.685	518.774
s4	177.0642	161.476	1.097	0.273	-140.313	494.442
s5	751.2793	171.902	4.370	0.000	413.409	1089.150
s6	67.6254	65.984	1.025	0.306	-62.065	197.316

Omnibus:	1.506	Durbin-Watson:	2.029
Prob(Omnibus):	0.471	Jarque-Bera (JB):	1.404
Skew:	0.017	Prob(JB):	0.496
Kurtosis:	2.726	Cond. No.	227.

- Regression equation

DP(Disease Progression) = 152.134 - 10.012 * age - 239.819 * sex + 519.840 * bmi + 324.390 * bp - 792.184 * s1 + 476.746 * s2 + 101.045 * s3 + 177.064 * s4 + 751.279 * s5 + 67.625 * s6

- Regression prediction and error

patient no.	DP predict	DP real	error
0	206.117070	151.0	0.365
1	68.072348	75.0	0.092
2	176.884060	141.0	0.254
3	166.917966	206.0	0.190
4	128.459842	135.0	0.048

3. 중고차 데이터를 이용한 Ridge regression

kaggle.com의 100,000 UK Used Car Data set 중 bmw 사의 중고차 데이터를 활용하여 lasso 회귀분석을 진행하였다. 중고차 데이터는 차 가격, 연식, 모델명, 급유 타입 등 9개의 feature data로 이루어져 있다. 주어진 feature data를 lasso 모형으로 분석해보았다.

- Ridge coefficient

```
[ 4274.03396413 -2836.85882873  -657.84291184 -3747.86954999
  2019.80422065 -1739.66443689 -1457.52331014  -793.5496383
  -563.97074491    37.23187096    70.36165581   884.00876313
  1664.5755971    360.49802416   608.00006044  1257.79064917
  1252.59711754   214.94396995  -563.82913075  -142.2577761
   787.1896652    648.04244403  2393.20873209  1424.9870521
  2546.84740961   517.16765466   103.33553591  2846.79179372
  1397.88915576  -100.4994292   -277.46758126   333.77352255
  -136.45750735   129.29666283  1465.14717823   405.72753599
  -434.6272607 ]
```

- R square 모형 평가값

0.8582480408503247

- MSE 모형 평가값

4189.327685966202

4. python code

① logistic regression

```
from sklearn import datasets
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split

from sklearn.linear_model import LogisticRegression

from sklearn.model_selection import cross_val_score

from sklearn.metrics import precision_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report

import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# 데이터 불러오기
raw_iris = datasets.load_iris()

# 피쳐, 타겟 데이터 지정
X = raw_iris.data
y = raw_iris.target

# target label 값을 0과 0 이외로 치환
y = np.where(y >= 1, 1, y)
print(y)

print(X.shape)

print(pd.DataFrame(X).head())

# 트레이닝/테스트 데이터 분할
X_tn, X_te, y_tn, y_te=train_test_split(X,y,random_state=1)

# 데이터 표준화
std_scale = StandardScaler()
std_scale.fit(X_tn)
```

```
X_tn_std = std_scale.transform(X_tn)
X_te_std = std_scale.transform(X_te)

print(X_tn_std.shape)   # 데이터 크기 확인
print(y_tn.shape)
print(pd.DataFrame(y_tn).head())  # y 레이블 확인

feature_id = 3  # index of feature
patient_num = 50 # first number of iris

plt.figure(1, [10, 5])
plt.subplot(121)
plt.plot(np.linspace(0, 4, 5), X_tn_std[0:5, feature_id], marker = "x")
plt.title('Feature plot', fontsize = 20)

plt.subplot(122)
plt.scatter(X_tn_std[0:patient_num, feature_id], y_tn[0:patient_num])
plt.title('First ' + str(patient_num) + ' iris for ' + str(feature_id) +
'-th feature', fontsize = 20)

plt.show()

x = X_tn_std[0:patient_num, feature_id]
y = y_tn[0:patient_num]

print(y)

float_epsilon = np.finfo(float).eps
Y = np.log(y/(1 - y + float_epsilon) + float_epsilon)
print(Y)

A = [[np.sum(x * x), np.sum(x)], [np.sum(x), len(x)]]
b = [np.sum(Y * x), np.sum(Y)]

U = np.linalg.solve(A, b)

print(U)

W = U[0]
```

```
b = U[1]

plt.figure(2, [10, 5])
plt.subplot(121)
plt.scatter(x, y)

plt.subplot(121)
x_ = np.linspace(-2, 2, 20)
plt.plot(x_, 1 / (1 + np.exp(-W * x_ - b)), 'r') # logistic app. sol.

plt.title('Original plot for iris data')

plt.subplot(122)
plt.scatter(x, 1 / (1 + np.exp(-W * x - b)))

plt.subplot(122)
x_ = np.linspace(-2, 2, 20)
plt.plot(x_, 1 / (1 + np.exp(-W * x_ - b)), 'r') # logistic app. sol.

plt.title('Logistic plot for iris data')

plt.show()

clf_logi_l2 =  LogisticRegression(penalty='l2')
clf_logi_l2.fit(X_tn_std, y_tn)
pred_logistic = clf_logi_l2.predict(X_te_std)
pred_proba = clf_logi_l2.predict_proba(X_te_std)
precision = precision_score(y_te, pred_logistic)
conf_matrix = confusion_matrix(y_te, pred_logistic)
class_report = classification_report(y_te, pred_logistic)
```

② OLS regression

```
from sklearn import datasets

import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import statsmodels.api as sm
```

```
raw_diabetes = datasets.load_diabetes()

dfX0 = pd.DataFrame(raw_diabetes.data, columns = raw_diabetes.feature_nam
es)
dfX = sm.add_constant(dfX0)
dfy = pd.DataFrame(raw_diabetes.target, columns=["DP"])

df = pd.concat([dfX, dfy], axis = 1)
df.tail()

model = sm.OLS(dfy, dfX)
result = model.fit()

print(result.summary())

result.predict(dfX.head())

print(df.head())
```

③ OLS regression

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import statsmodels.api as sm

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import Ridge
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error

raw_cars = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/bmw.csv')

df = pd.get_dummies(data = raw_cars, columns = ["model", "transmission",
"fuelType"])

X = df.drop("price", axis = 1)
y = df["price"]
```

```
X_tn, X_te, y_tn, y_te = train_test_split(X, y, random_state = 1)

std_scale = StandardScaler()
std_scale.fit(X_tn)
X_tn_std = std_scale.transform(X_tn)
X_te_std = std_scale.transform(X_te)

clf_Ridge = Ridge(alpha = 0.01)
clf_Ridge.fit(X_tn_std, y_tn)

print(clf_Ridge.coef_)
print(clf_Ridge.intercept_)

pred_Ridge = clf_Ridge.predict(X_te_std)

print(r2_score(y_te, pred_Ridge))

print(np.sqrt(mean_squared_error(y_te, pred_Ridge)))
```