

Lab6

November 14, 2025

1 Lab 6

1.1 Normalisation

```
[1]: from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
cancer = load_breast_cancer()

X_train, X_test, y_train, y_test = train_test_split(cancer.data, cancer.target,random_state=42)
print(X_train.shape)
print(X_test.shape)
```

(426, 30)

(143, 30)

```
[2]: from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
```

```
[3]: scaler.fit(X_train)
```

```
[3]: MinMaxScaler()
```

```
[4]: X_train_scaled = scaler.transform(X_train)
print(X_train_scaled.shape)
print(X_train.min(axis=0))
print(X_train.max(axis=0))
print(X_train_scaled.min(axis=0))
print(X_train_scaled.max(axis=0))
```

(426, 30)

```
[7.691e+00 9.710e+00 4.792e+01 1.704e+02 5.263e-02 1.938e-02 0.000e+00
 0.000e+00 1.167e-01 4.996e-02 1.115e-01 3.602e-01 7.570e-01 6.802e+00
 1.713e-03 2.252e-03 0.000e+00 0.000e+00 7.882e-03 8.948e-04 8.678e+00
 1.202e+01 5.449e+01 2.236e+02 7.117e-02 2.729e-02 0.000e+00 0.000e+00
 1.565e-01 5.504e-02]
[2.811e+01 3.928e+01 1.885e+02 2.501e+03 1.634e-01 3.114e-01 4.268e-01
 2.012e-01 3.040e-01 9.744e-02 2.873e+00 4.885e+00 2.198e+01 5.422e+02
 3.113e-02 1.354e-01 3.960e-01 5.279e-02 6.146e-02 2.984e-02 3.604e+01
```

```

4.954e+01 2.512e+02 4.254e+03 2.184e-01 9.379e-01 9.608e-01 2.910e-01
6.638e-01 1.730e-01]
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0.]
[1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1. 1. 1. 1. 1.]
```

[5]:

```
X_test_scaled = scaler.transform(X_test)
print(X_test_scaled.min(axis=0))
print(X_test_scaled.max(axis=0))
```

```

[-0.03477154  0.0226581 -0.02937829 -0.01154209  0.1185339  0.05547565
 0.          0.         -0.0571276   0.05686605  0.00184682  0.00057461
 0.00067851  0.00402131  0.04949519  0.02556554  0.          0.
 0.03092687  0.01120048 -0.02733718  0.01252665 -0.02074119 -0.00952759
 0.11424302  0.03036426  0.          0.          0.00019712  0.03399457]
[0.85846516  0.72404464  0.87907241  0.73268686  0.76257109  1.11643038
 0.87956888  0.91699801  0.92845702  0.68386689  0.42712294  0.7814268
 0.41831975  0.36028898  0.48703131  0.78219725  0.76717172  0.62928585
 1.32643996  0.75885466  0.87025802  0.93656716  0.81088913  0.79605002
 1.02852679  1.13188961  1.30308077  0.9975945  0.76384782  1.29247202]
```

[6]:

```
from sklearn.svm import SVC
svm = SVC(C=1)
svm.fit(X_train, y_train)
print(svm.score(X_test, y_test))
```

0.951048951048951

[7]:

```
X_test_scaled = scaler.transform(X_test)
svm.fit(X_train_scaled, y_train)
print(svm.score(X_test_scaled, y_test))
```

0.9790209790209791

1.2 Parameter Selection using a Validation Set and Cross-Validation

[8]:

```
X_train, X_test, y_train, y_test = train_test_split(cancer.data, cancer.target, random_state=42)
best_score = 0
for gamma in [0.001, 0.01, 0.1, 1, 10, 100]:
    for C in [0.001, 0.01, 0.1, 1, 10, 100]:
        svm = SVC(gamma=gamma, C=C)
        svm.fit(X_train, y_train)

        score = svm.score(X_test, y_test)

        if score > best_score:
            best_score = score
```

```

        best_C = C
        best_gamma = gamma
print("Best score:", best_score)
print("Best parameters C and gamma:", best_C, best_gamma)

```

Best score: 0.9230769230769231
 Best parameters C and gamma: 1 0.001

```
[9]: X_train, X_test, y_train, y_test = train_test_split(cancer.data, cancer.target, □
    ↵random_state=42)
X_train_pr, X_valid, y_train_pr, y_valid = train_test_split(X_train, y_train, □
    ↵random_state=42)
print("Sizes of train_pr, valid, and test sets:",
X_train_pr.shape[0], X_valid.shape[0], X_test.shape[0])
best_score = 0
for gamma in [0.001, 0.01, 0.1, 1, 10, 100]:
    for C in [0.001, 0.01, 0.1, 1, 10, 100]:
        svm = SVC(gamma=gamma, C=C)
        svm.fit(X_train_pr, y_train_pr)

        score = svm.score(X_valid, y_valid)

        if score > best_score:
            best_score = score
            best_C = C
            best_gamma = gamma

svm = SVC(C=best_C, gamma=best_gamma)
svm.fit(X_train, y_train)
test_score = svm.score(X_test, y_test)

print("Best score on validation set:", best_score)
print("Best parameters C and gamma:", best_C, best_gamma)
print("Test set score with best parameters:", test_score)
```

Sizes of train_pr, valid, and test sets: 319 107 143
 Best score on validation set: 0.8878504672897196
 Best parameters C and gamma: 1 0.001
 Test set score with best parameters: 0.9230769230769231

```
[10]: import numpy as np
from sklearn.model_selection import cross_val_score
best_score = 0
for gamma in [0.001, 0.01, 0.1, 1, 10, 100]:
    for C in [0.001, 0.01, 0.1, 1, 10, 100]:
        svm = SVC(gamma=gamma, C=C)
        scores = cross_val_score(svm, X_train, y_train, cv=5)
        score = np.mean(scores)
```

```

        if score > best_score:
            best_score = score
            best_C = C
            best_gamma = gamma

svm = SVC(C=best_C, gamma=best_gamma)
svm.fit(X_train, y_train)
test_score = svm.score(X_test, y_test)
print("Best CV score:", best_score)
print("Best parameters C and gamma:", best_C, best_gamma)
print("Test set score with best parameters:", test_score)

```

Best CV score: 0.9177838577291382
 Best parameters C and gamma: 1 0.001
 Test set score with best parameters: 0.9230769230769231

[11]: param_grid = {'C': [0.001, 0.01, 0.1, 1, 10, 100], 'gamma': [0.001, 0.01, 0.1, 1, 10, 100]}

[12]: from sklearn.model_selection import GridSearchCV
 grid_search = GridSearchCV(SVC(), param_grid, cv=5)

[13]: X_train, X_test, y_train, y_test = train_test_split(cancer.data, cancer.target, random_state=42)

[14]: grid_search.fit(X_train, y_train)

[14]: GridSearchCV(cv=5, estimator=SVC(),
 param_grid={'C': [0.001, 0.01, 0.1, 1, 10, 100],
 'gamma': [0.001, 0.01, 0.1, 1, 10, 100]})

[15]: grid_search.score(X_test, y_test)

[15]: 0.9230769230769231

[16]: print(grid_search.best_params_)

print(grid_search.best_score_)

{'C': 1, 'gamma': 0.001}
 0.9177838577291382

1.3 Exercises

[17]: X = np.genfromtxt("diabetes.data", delimiter="\t", skip_header=1, usecols=np.arange(10))
 y = np.genfromtxt("diabetes.data", delimiter="\t", skip_header=1, usecols=10)

 print(X[:3])

```
print(y[:3])  
  
[[ 59.      2.      32.1     101.     157.     93.2     38.      4.  
    4.8598   87.      ]]  
[ 48.      1.      21.6     87.      183.     103.2     70.      3.  
    3.8918   69.      ]]  
[ 72.      2.      30.5     93.      156.     93.6     41.      4.  
    4.6728   85.      ]]  
[151.    75.    141.]
```

```
[18]: X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=2408)
```

```
[19]: from sklearn.linear_model import Lasso  
lasso = Lasso().fit(X_train, y_train)  
print("R^2:      ", lasso.score(X_test,y_test))  
print("Features: ", np.sum(lasso.coef_ != 0))
```

```
R^2:      0.5797150477335155  
Features: 10
```

```
[20]: 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)  
  
lasso.fit(X_train_scaled, y_train)  
print("R^2:      ", lasso.score(X_test_scaled,y_test))  
print("Features: ", np.sum(lasso.coef_ != 0))
```

```
R^2:      0.5813828475346058  
Features: 8
```

```
[21]: best_score = 0  
for alpha in [0.001, 0.01, 0.1, 1, 10, 100]:  
    lasso = Lasso(alpha=alpha)  
    scores = cross_val_score(lasso, X_train_scaled, y_train, cv=5)  
    score = np.mean(scores)  
  
    if score > best_score:  
        best_score = score  
        best_alpha = alpha  
  
lasso = Lasso(alpha=best_alpha)  
lasso.fit(X_train_scaled, y_train)  
test_score = lasso.score(X_test_scaled, y_test)  
print("Best CV score:", best_score)  
print("Best parameters C and gamma:", best_alpha)
```

```
print("Test set score with best parameters:", test_score)
```

```
Best CV score: 0.4404470919980853
Best parameters C and gamma: 0.01
Test set score with best parameters: 0.5803663929881258
```