Artificial Intelligence PBL-1

Classifier for MNIST with LR, and SVM

Team 4

Given Problem

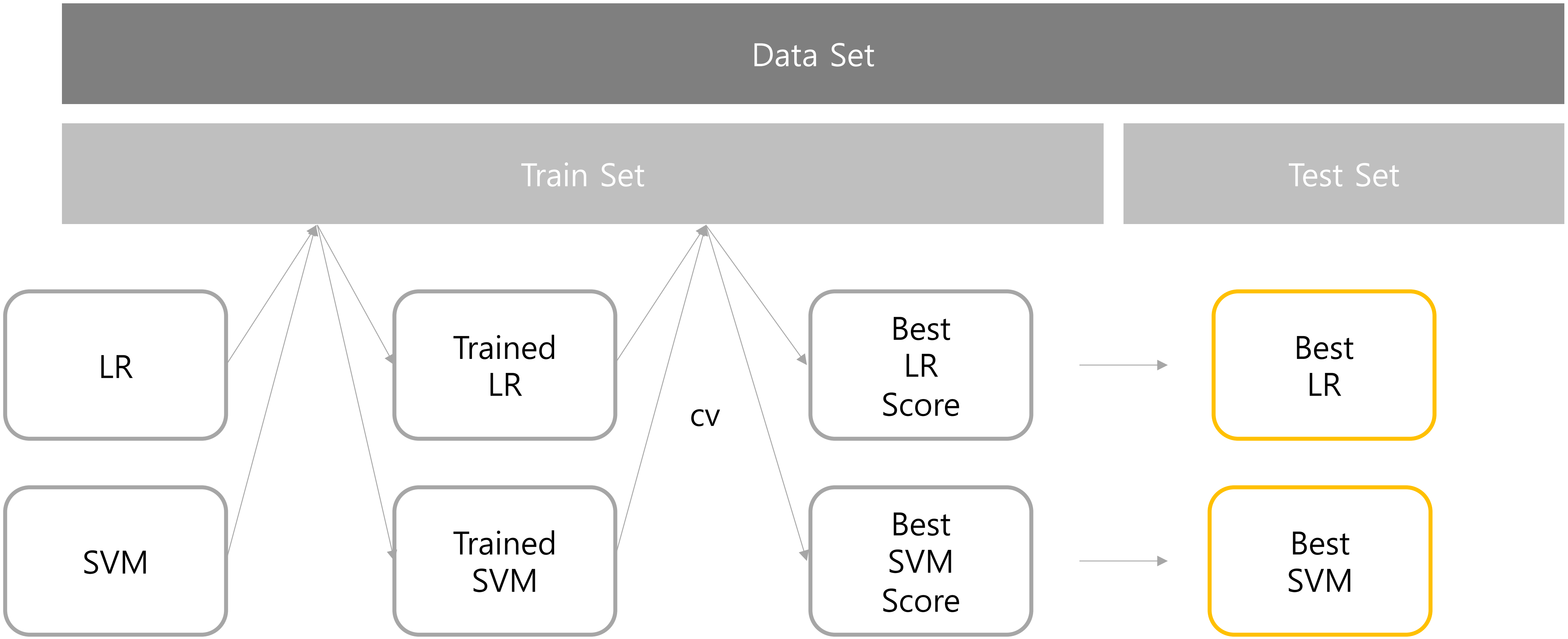
- Problem : making an automated zipcode recognition system

- Main objective:

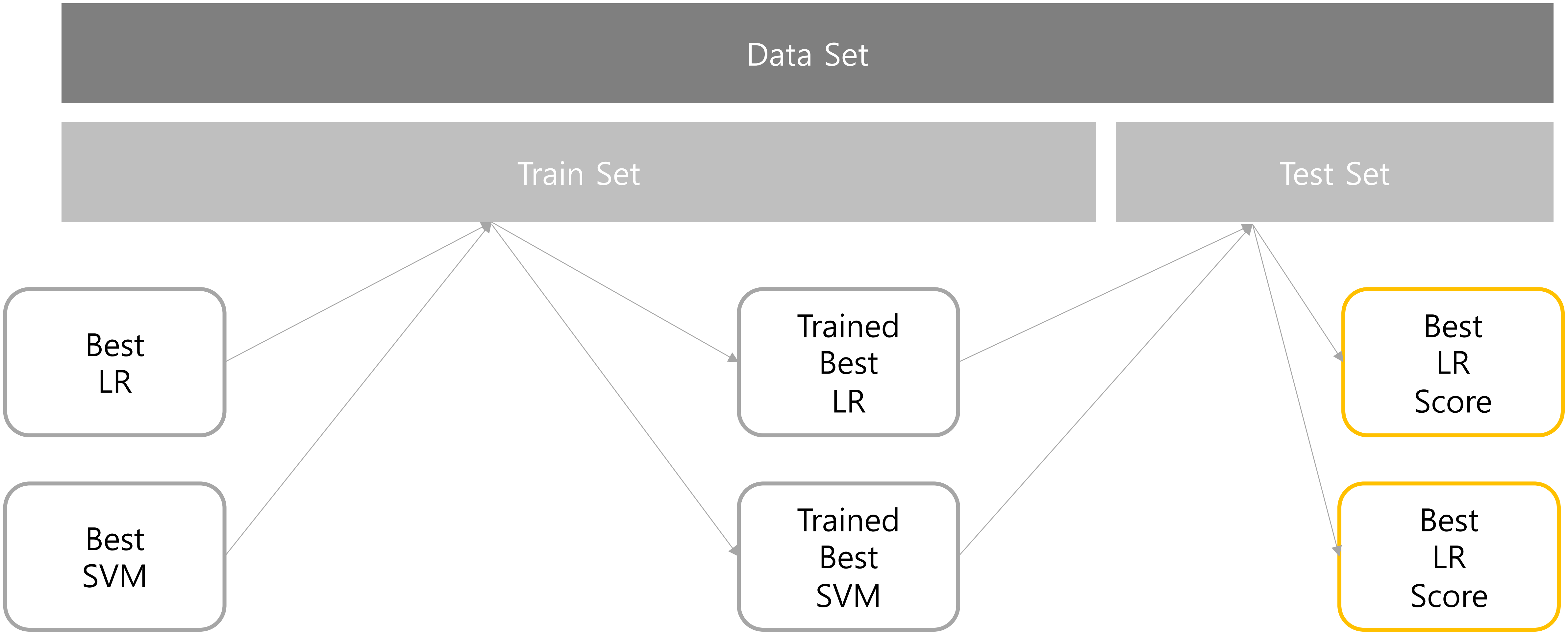
  Find the best model out of the following settings:

  - Classifiers: Logistic Regression and SVM
  - Data: 70k number(0~9) images of MNIST
  - Criterion:
    - Model selection: f1-score(micro)
    - Test performance: f1-score(micro) of the final model on the test images

Solution


Data Set
Train Set
Test Set
LR
SVM
Trained LR
Trained SVM
cv
Best LR Score
Best SVM Score
Best LR
Best SVM

Solution


Data Set
Train Set
Test Set
Best LR
Best SVM
Trained Best LR
Trained Best SVM
Best LR Score
Best LR Score

Solution

Dataset

train_test_split()

StandardScaler()

deskew()

LR

LogisticRegression()

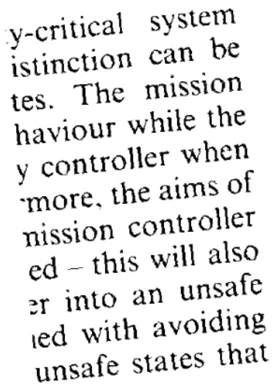
SVM

SVC()

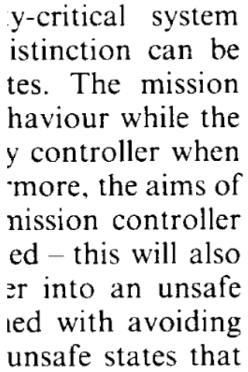
Train&Valid

GridSearchCV()

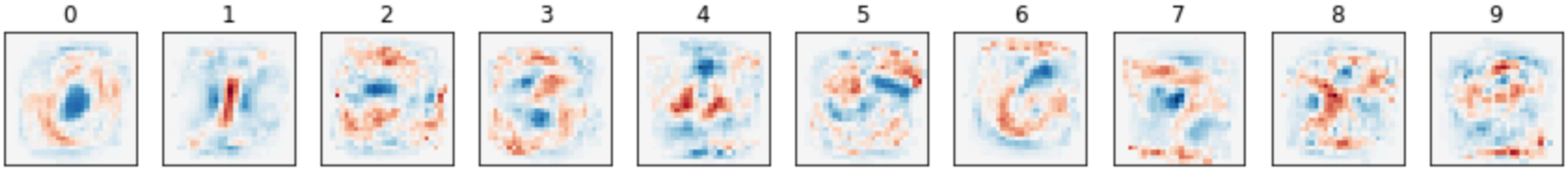
deskew()


y-critical system
istinction can be
tes. The mission
haviour while the
y controller when
more, the aims of
nission controller
ed – this will also
er into an unsafe
ıed with avoiding
unsafe states that

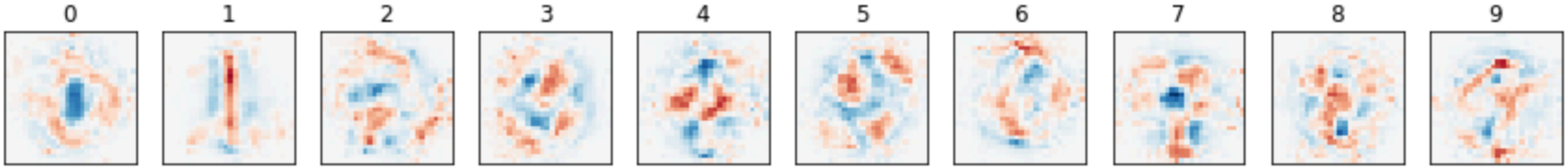
deskew


y-critical system
istinction can be
tes. The mission
haviour while the
y controller when
more, the aims of
nission controller
ed – this will also
er into an unsafe
ıed with avoiding
unsafe states that

do not(acc : 0.944375)



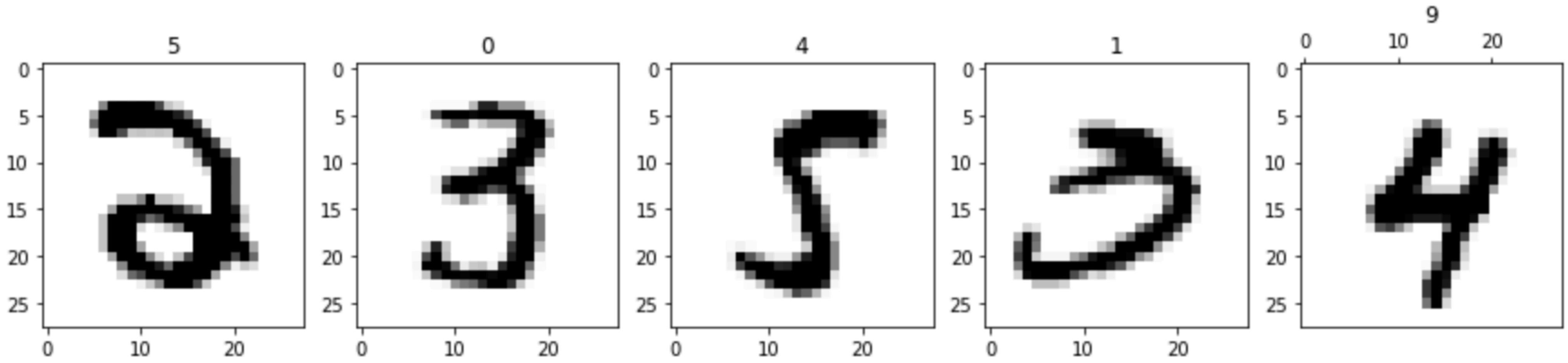
do(acc : 0.952)



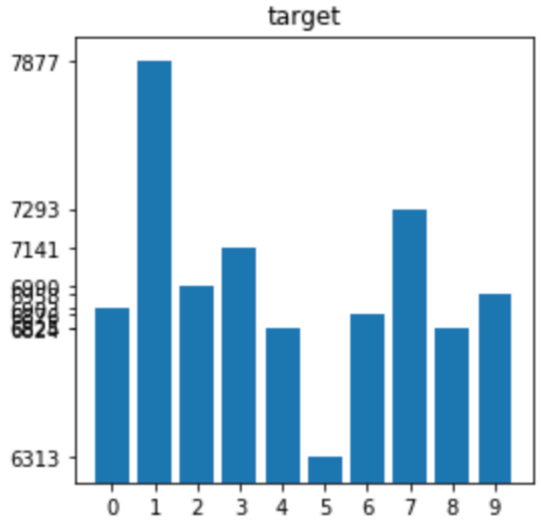
Logistic Regression C: 0.01, penalty: l2, solver: lbfgs

MNIST DATA

- 28x28 pixel
- pixel value range [0, 255]
- 70000 samples
- 786 features
- many pixel values are zero
- large '1's, small '5's
- no outlier, missing value



from sklearn.model_selection import train_test_split

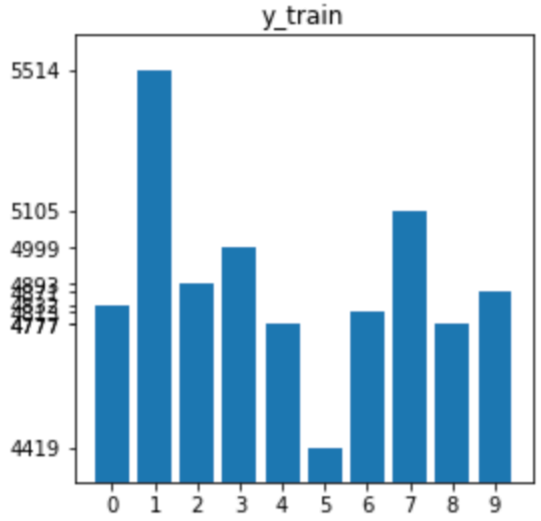


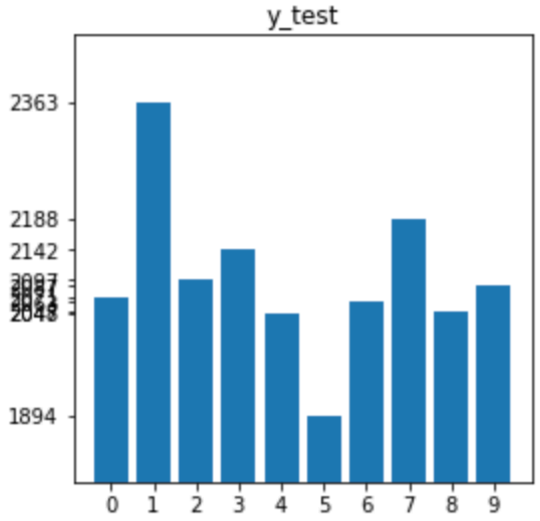
X_train : (49000, 784)

X_test : (21000, 784)

y_train : (49000,)

y_test : (21000,)






```
mnist = fetch_openml("mnist_784")
X = mnist.data
y = mnist.target

X_train, X_test, y_train, y_test
= train_test_split(X, y, test_size=0.3,
                   random_state=123, stratify=y)
```

Dataset

from sklearn.preprocessing import StandardScaler

```
scaler = StandardScaler()
scaler.fit(X_train)
X_train = scaler.transform(X_train)
X_test = scaler.transfrom(X_test)
```

Scaler:

StandardScaler() # mean: 0, std: 1

MinMaxScaler() # max: 1, min: 0

RobustScaler() # median: 0, IQR: 1

Standard score : $\frac{X - \mu}{\sigma}$

Min-Max Feature scaling : $\frac{X - X_{min}}{X_{max} - X_{min}}$

GridSearchCV

from sklearn.model_selection import GridSearchCV

scoring = **f1_micro**

cv = 10

$$F_1 = \left(\frac{2}{recall^{-1} + precision^{-1}}\right) = 2 \cdot \left(\frac{precision \cdot recall}{precision + recall}\right)$$

- f1_score – macro : Calculate metrics for each label, and find their unweighted mean. This does not take label imbalance into account.

  – micro : Calculate metrics globally by counting the total true positives, false negatives and false positives.

- cv : Determines the cross-validation splitting strategy.

1	2	3	4	5	6	7	8	9	10

Models

Logistic Regression

```
parameters = {'solver':['saga'], 'penalty':['l1', 'l2'], 'C':[100.0, 10.0, 1.0, 0.1, 0.01]}

lc = LogisticRegression(multi_class='ovr')
clf = GridSearchCV(estimator=lc, param_grid=parameters, scoring='f1_micro', cv=10, n_jobs=-1)
clf.fit(X_train, y_train)
```

Support Vector Machine

```
parameters = {'kernel':['linear'], 'C':[100.0, 10.0, 1.0, 0.1, 0.01, 0.001]}

svc = SVC()
clf = GridSearchCV(estimator=svc, param_grid=parameters, scoring='f1_micro', cv=10, n_jobs=-1)
clf.fit(X_train, y_train)
```

LogisticRegression

from sklearn.linear_model import LogisticRegression

LogisticRegression() : Logistic Regression classifier.

- multi_class : ovr(one-vs-rest)
- penalty : l1, l2

$$\| x \|_p := (\sum_{i=1}^{n} |x_i|^p)^{1/p}$$

if p = 1, l1
if p = 2, l2

- C : 100.0, 10.0, 1.0, 0.1, 0.01
- solver : saga

$$w^{k+1} = w^k - \alpha(f_j'(w^k) - f_j'(\theta_j^k) + \frac{1}{n}\sum_{i=1}^{n} f_i'(\theta_i^k))$$

▶ best parameter :

penalty = l2
C = 100.0
solver = saga

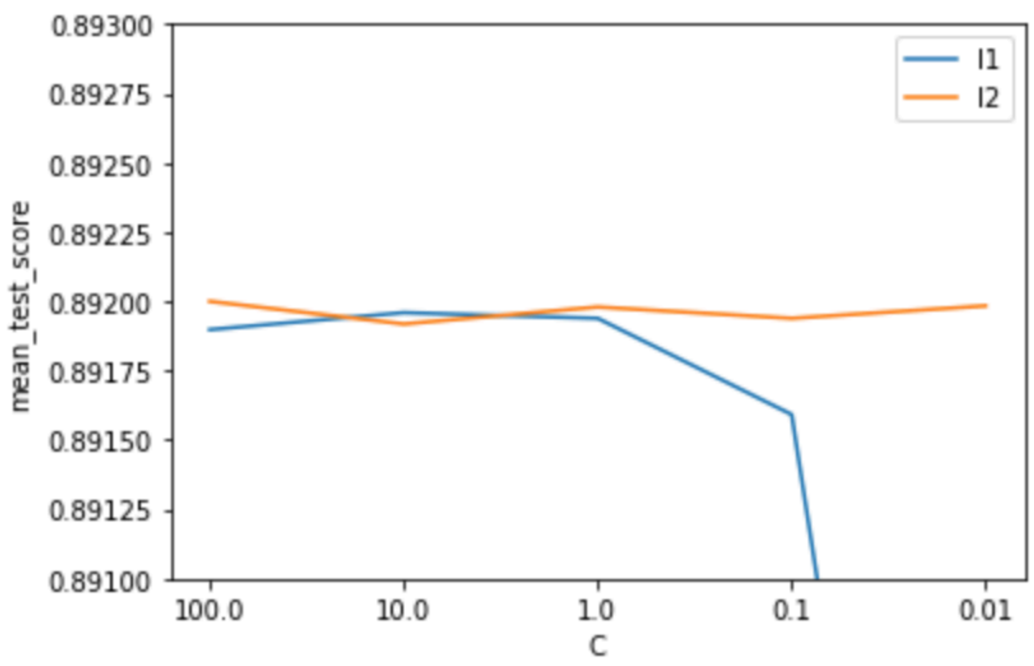
▶ best score :

f1_micro = 0.892

Validation

- Logistic Regression

C	f1	mean_test_score	mean_fit_time(min)
100.0	l1	0.89189796	14
	l2	0.892	10
10.0	l1	0.89195918	15
	l2	0.89191837	10
1.0	l1	0.89193878	15
	l2	0.89197959	10
0.1	l1	0.89159184	16
	l2	0.89193878	10
0.01	l1	0.88708163	14
	l2	0.8919837	11



Intel Xeon, RAM 13GB

from sklearn.svm import SVC

SVC() : C-Support Vector Classification

- kernel : linear
- C : 100.0, 10.0, 1.0, 0.1, 0.01, 0.001

▶ best parameter :

C = 0.01

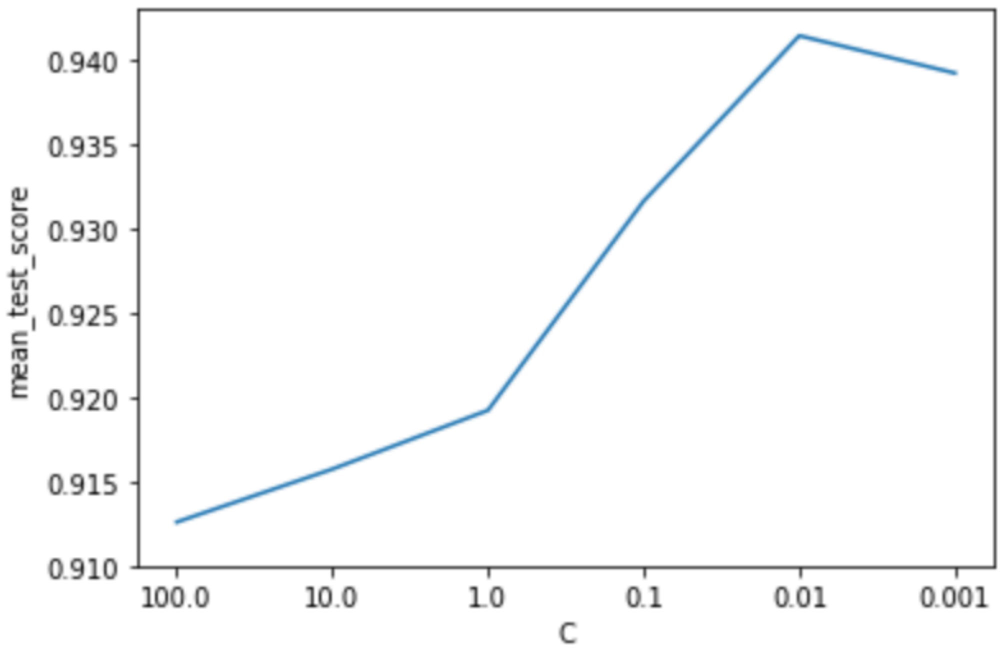
▶ best score :

f1_micro = 0.94155102

Validation

- SVM

C \ f1	mean_test_score	mean_fit_time(min)
100.0	0.91259184	215
10.0	0.91573469	23
1.0	0.91922449	10
0.1	0.93163265	9
0.01	0.94144102	9
0.001	0.93922449	11



AMD 2700x, RAM 32GB

Test

LR

multi_class = ovr

penalty = l2

C = 100.0

solver = saga

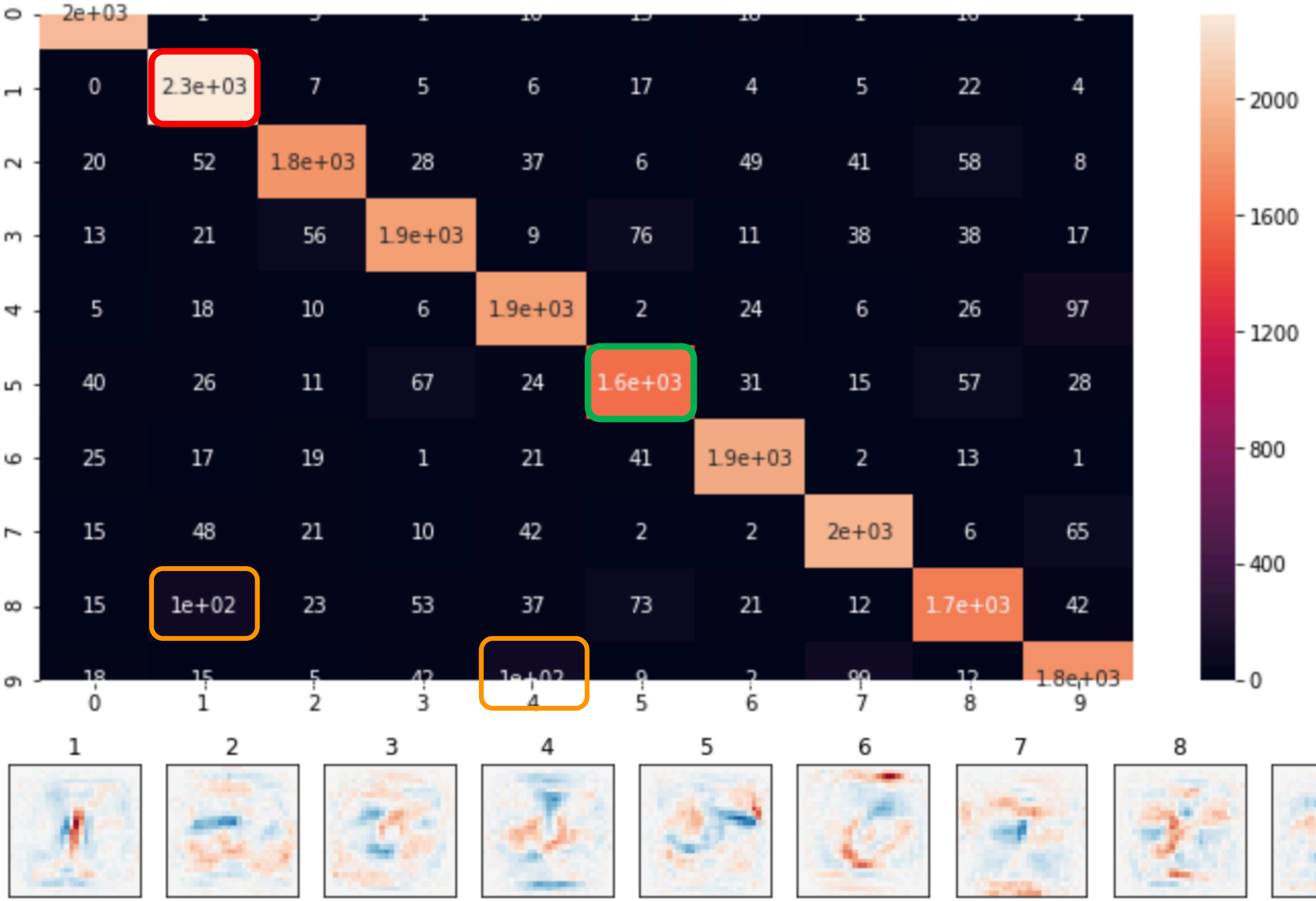
SV
M

kernel = linear

C = 0.01

Test

- LR



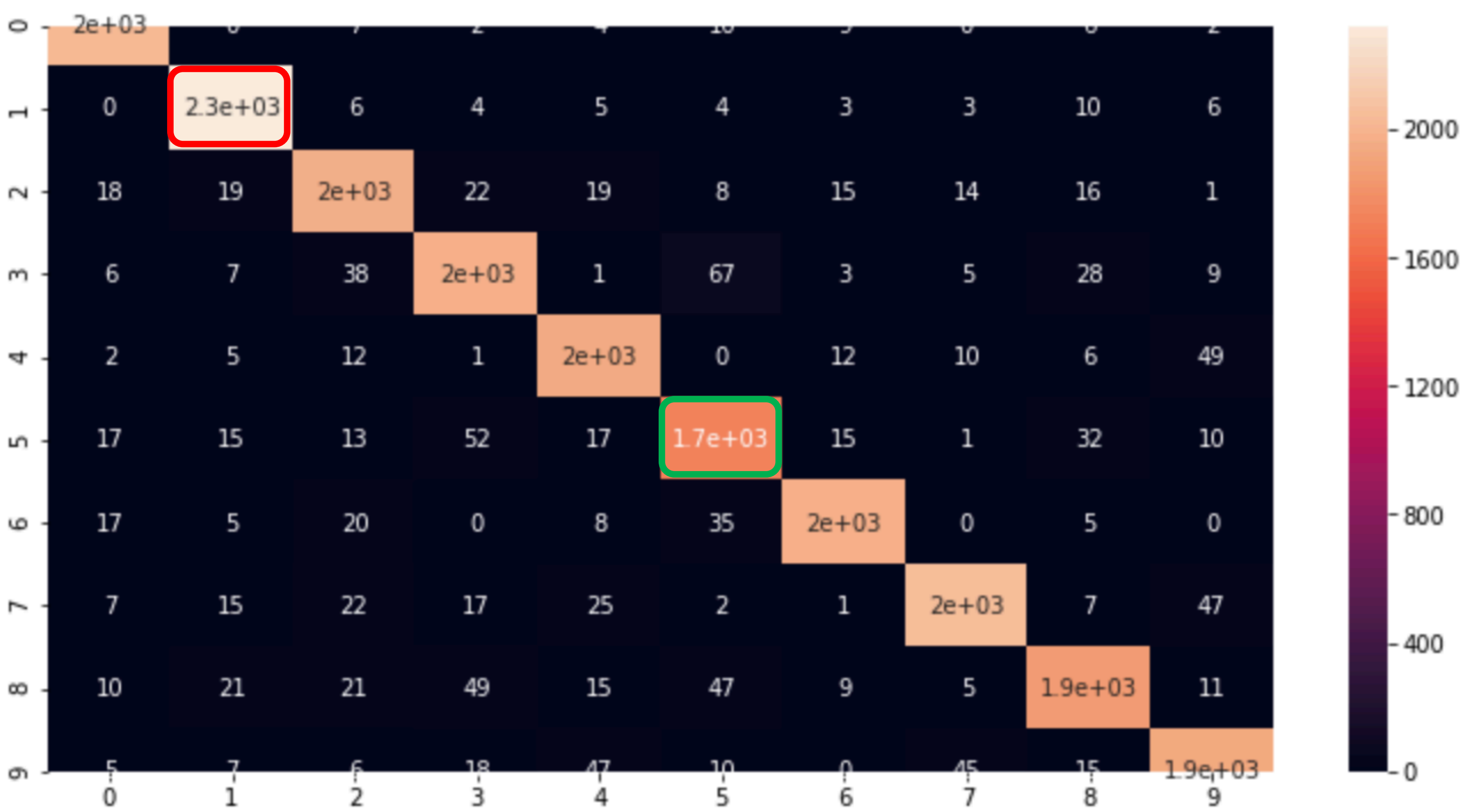
Test

- LR

	precision	recall	f1-score	support
0	0.93	0.97	0.95	2071
1	0.88	0.97	0.93	2363
2	0.92	0.86	0.89	2097
3	0.90	0.87	0.88	2142
4	0.87	0.90	0.89	2047
5	0.87	0.84	0.86	1894
6	0.92	0.93	0.93	2063
7	0.90	0.90	0.90	2188
8	0.87	0.82	0.84	2048
9	0.87	0.86	0.86	2087
micro avg	0.89	0.89	0.89	21000
macro avg	0.89	0.89	0.89	21000
weighted avg	0.89	0.89	0.89	21000

Test

- SVM


2e+03
2.3e+03
2e+03
2e+03
2e+03
1.7e+03
2e+03
2e+03
1.9e+03
1.9e+03
0 1 2 3 4 5 6 7 8 9
2000
1600
1200
800
400
0

Test

- SVM

	precision	recall	f1-score	support
0	0.96	0.98	0.97	2071
1	0.96	0.98	0.97	2363
2	0.93	0.94	0.93	2097
3	0.92	0.92	0.92	2142
4	0.93	0.95	0.94	2047
5	0.90	0.91	0.91	1894
6	0.97	0.96	0.96	2063
7	0.96	0.93	0.95	2188
8	0.94	0.91	0.92	2048
9	0.93	0.93	0.93	2087
micro avg	0.94	0.94	0.94	21000
macro avg	0.94	0.94	0.94	21000
weighted avg	0.94	0.94	0.94	21000

Test

- micro

	Accuracy	Precision	Recall	F1-score
LR	0.893619	0.893619	0.893619	0.893619
SVM	0.941905	0.941905	0.941905	0.941905

- macro

	Accuracy	Precision	Recall	F1-score
LR	0.893524	0.89332	0.892042	0.892141
SVM	0.941905	0.941301	0.941153	0.941147

$$ACC = \frac{TP + TN}{ALL} \qquad PRE_{micro} = \frac{TP_1 + \cdots + TP_k}{TP_1 + \cdots + TP_k + FP_1 + \cdots + FP_k} \qquad REC_{micro} = \frac{TP_1 + \cdots + TP_k}{TP_1 + \cdots + TP_k + FN_1 + \cdots + FN_k} \qquad F1 = 2 \cdot \frac{PRE \times REC}{PRE + REC}$$

Question and Answer

Thank you