COSE362 Machine Learning
Assignment 3

181211

김현재 고미영 김병주

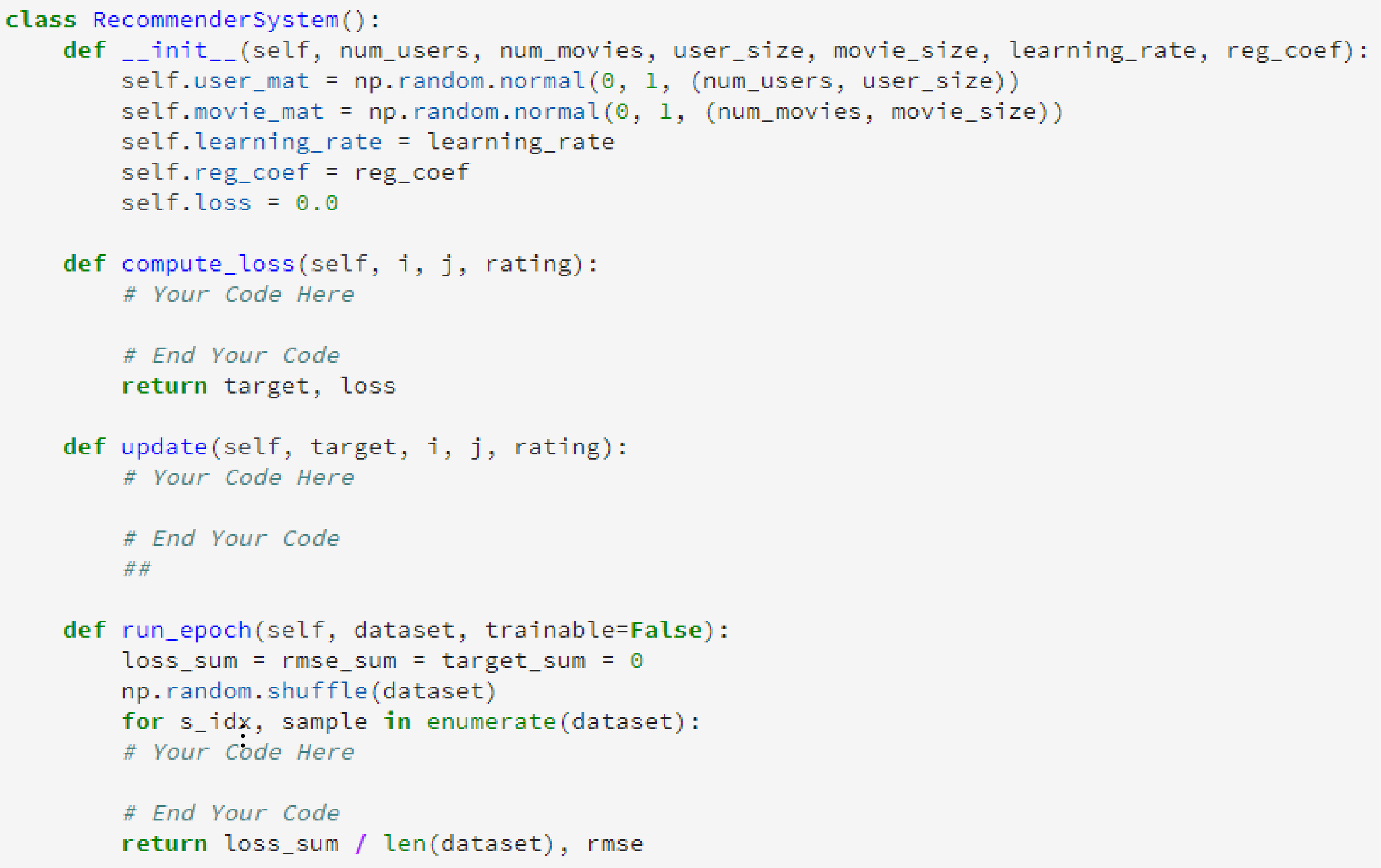
Data Mining & Information Systems Lab.
Department of Computer Science and Engineering,
College of Informatics, Korea University

Problem 1 – Recommender System

Movie	Alice (1)	Bob (2)	Carol (3)	Dave (4)	…
Love at last	5	5	0	0	
Romance forever	5	?	?	0	
Cute puppies of love	?	4	0	?	
Nonstop car chases	0	0	5	4	
Swords vs. karate	0	0	5	?	
⋮					

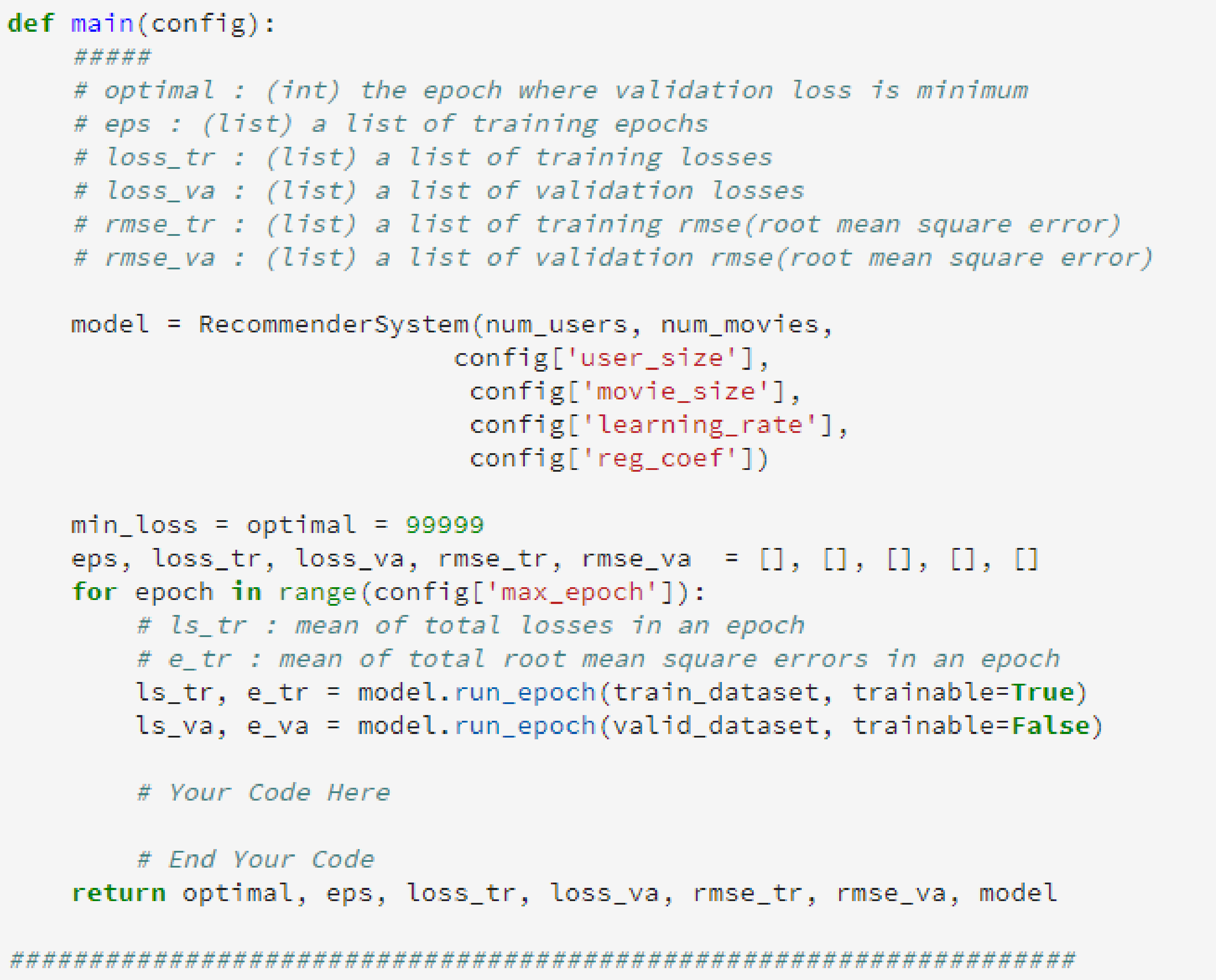
- Recommender System – Collaborative Filtering
  - ✓ # of users : 943
  - ✓ # of items : 1682

Problem 1 – Recommender System



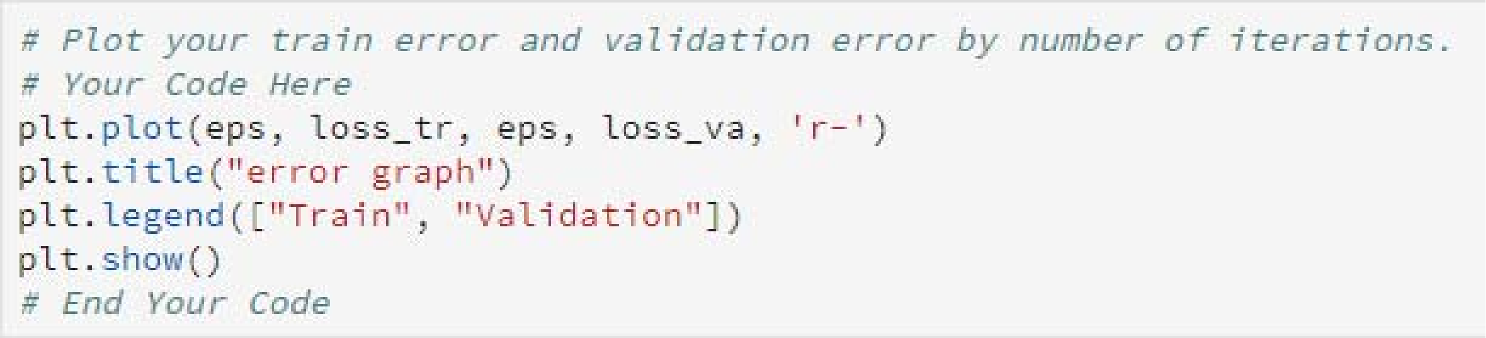
```python
class RecommenderSystem():
    def __init__(self, num_users, num_movies, user_size, movie_size, learning_rate, reg_coef):
        self.user_mat = np.random.normal(0, 1, (num_users, user_size))
        self.movie_mat = np.random.normal(0, 1, (num_movies, movie_size))
        self.learning_rate = learning_rate
        self.reg_coef = reg_coef
        self.loss = 0.0

    def compute_loss(self, i, j, rating):
        # Your Code Here

        # End Your Code
        return target, loss

    def update(self, target, i, j, rating):
        # Your Code Here

        # End Your Code
        ##

    def run_epoch(self, dataset, trainable=False):
        loss_sum = rmse_sum = target_sum = 0
        np.random.shuffle(dataset)
        for s_idx, sample in enumerate(dataset):
        # Your Code Here

        # End Your Code
        return loss_sum / len(dataset), rmse
```

Problem 1 – Recommender System

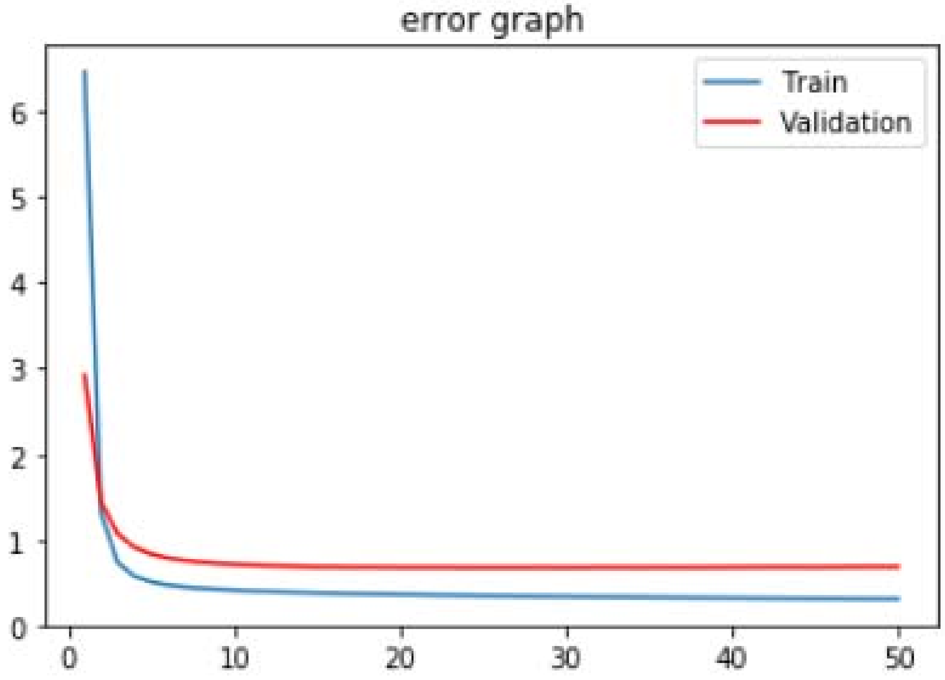


```
def main(config):
    #####
    # optimal : (int) the epoch where validation loss is minimum
    # eps : (list) a list of training epochs
    # loss_tr : (list) a list of training losses
    # loss_va : (list) a list of validation losses
    # rmse_tr : (list) a list of training rmse(root mean square error)
    # rmse_va : (list) a list of validation rmse(root mean square error)

    model = RecommenderSystem(num_users, num_movies,
                            config['user_size'],
                             config['movie_size'],
                             config['learning_rate'],
                             config['reg_coef'])

    min_loss = optimal = 99999
    eps, loss_tr, loss_va, rmse_tr, rmse_va  = [], [], [], [], []
    for epoch in range(config['max_epoch']):
        # ls_tr : mean of total losses in an epoch
        # e_tr : mean of total root mean square errors in an epoch
        ls_tr, e_tr = model.run_epoch(train_dataset, trainable=True)
        ls_va, e_va = model.run_epoch(valid_dataset, trainable=False)

        # Your Code Here

        # End Your Code
    return optimal, eps, loss_tr, loss_va, rmse_tr, rmse_va, model

###################################################################
```

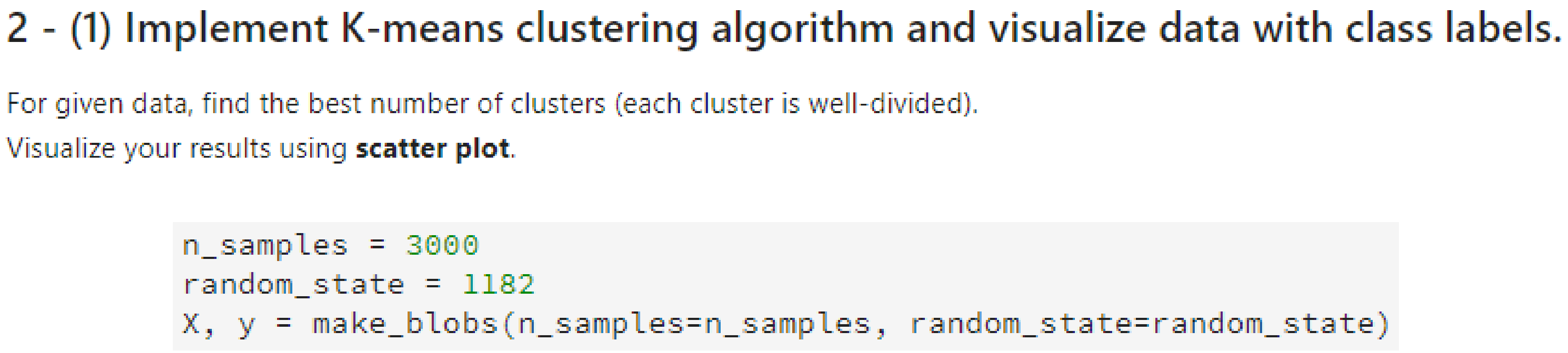
Problem 1 – Recommender System



```
# Plot your train error and validation error by number of iterations.
# Your Code Here
plt.plot(eps, loss_tr, eps, loss_va, 'r-')
plt.title("error graph")
plt.legend(["Train", "Validation"])
plt.show()
# End Your Code
```



Problem 2 - Clustering



2 - (1) Implement K-means clustering algorithm and visualize data with class labels.

For given data, find the best number of clusters (each cluster is well-divided).

Visualize your results using **scatter plot**.

```
n_samples = 3000
random_state = 1182
X, y = make_blobs(n_samples=n_samples, random_state=random_state)
```

- Unsupervised Learning - Clustering
  - ✓ # of samples : 3000

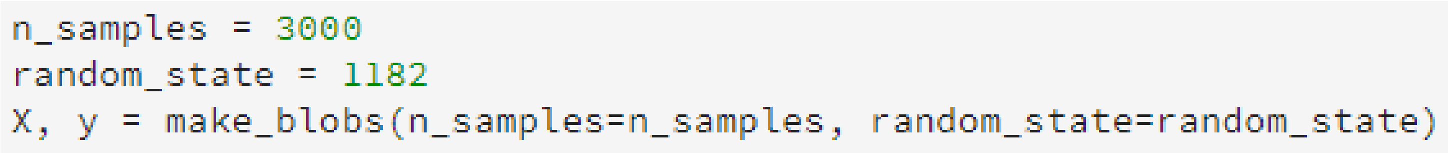
- For given data, find the best number of clusters(each cluster is well-divided).

Problem 2 - Clustering

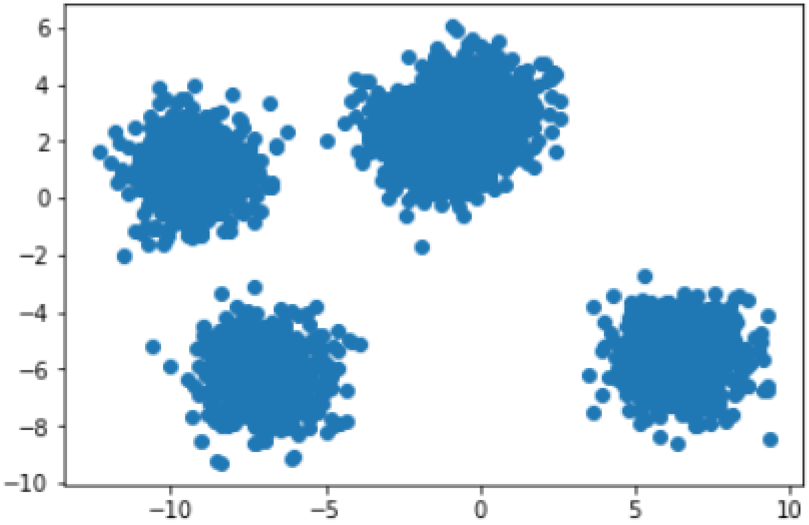
2 - (1) Implement K-means clustering algorithm and visualize data with class labels.

For given data, find the best number of clusters (each cluster is well-divided).

Visualize your results using **scatter plot**.

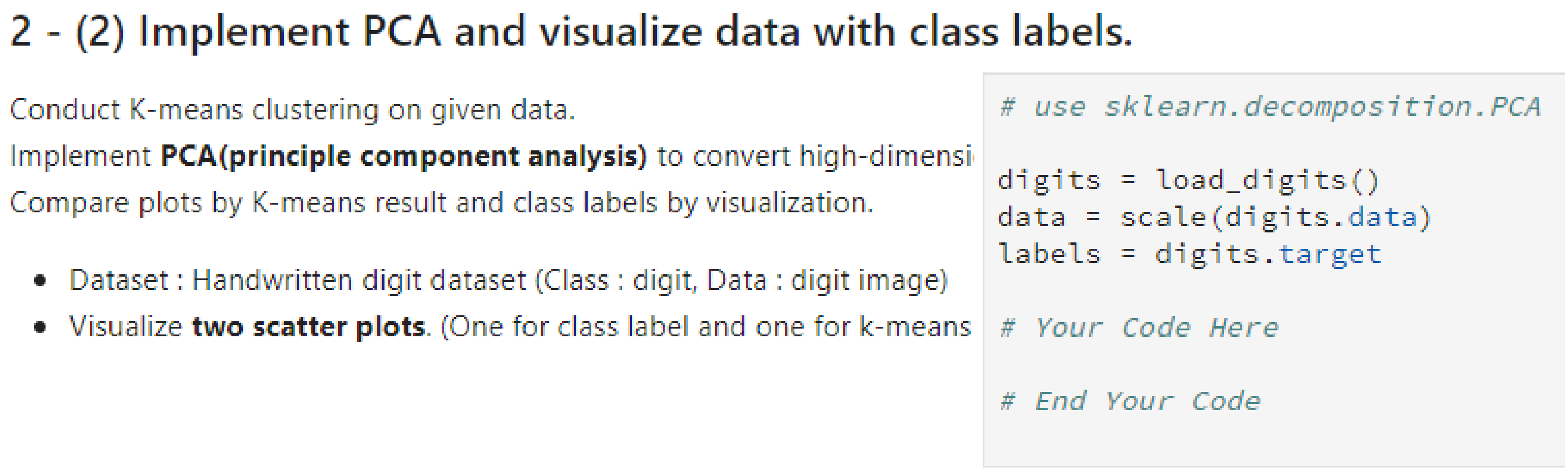


```
n_samples = 3000
random_state = 1182
X, y = make_blobs(n_samples=n_samples, random_state=random_state)
```



*** example**

Problem 2 - Clustering



2 - (2) Implement PCA and visualize data with class labels.

Conduct K-means clustering on given data.

Implement **PCA(principle component analysis)** to convert high-dimensi

Compare plots by K-means result and class labels by visualization.

- Dataset : Handwritten digit dataset (Class : digit, Data : digit image)
- Visualize **two scatter plots**. (One for class label and one for k-means

```
# use sklearn.decomposition.PCA

digits = load_digits()
data = scale(digits.data)
labels = digits.target

# Your Code Here

# End Your Code
```

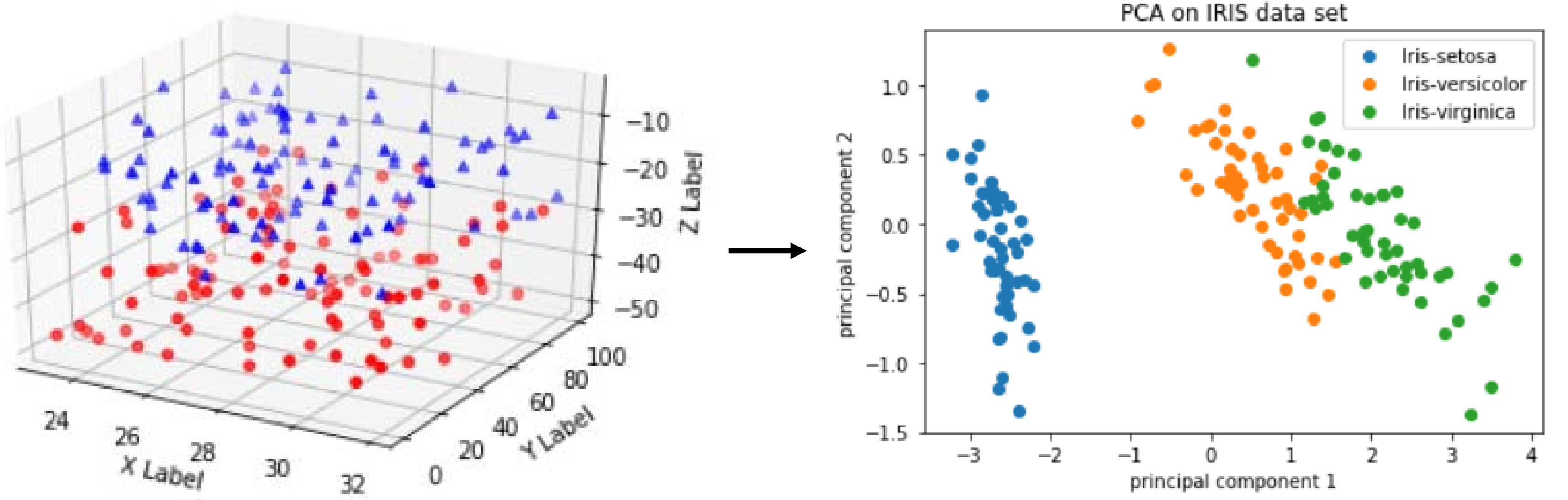
- Unsupervised Learning – PCA & Clustering
  - ✓ # of features : 64
  - ✓ # of samples : 1797
  - ✓ # of class : 10

Problem 2 - Clustering

2 - (2) Implement PCA and visualize data with class labels.

Conduct K-means clustering on given data.
Implement **PCA(principle component analysis)** to convert high-dimensional vectors into 2-dimensional vectors.
Compare plots by K-means result and class labels by visualization.

- Dataset : Handwritten digit dataset (Class : digit, Data : digit image)
- Visualize **two scatter plots**. (One for class label and one for k-means clustering)



*** example**