```python
from pathlib import Path
from collections import defaultdict

import numpy as np
import pandas as pd
from scipy import io
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix, accuracy_score
from sklearn.svm import SVC

from my_logger import get_my_logger

logger = get_my_logger(Path(__file__).name)


def load_data(data_name):
    path = Path(f'data/input/{data_name}_data.mat')
    data = io.loadmat(path)
    return data['training_data'], data['test_data'],
data['training_labels'].reshape(-1,)


def tr_va_split(x_train, y_train, n_va, shuffle=False):
    """
    x_train: np.array
        Features of all training dataset

    y_train: np.array
        Label of all training dataset

    n_va: float or int
        If float, shoud be 0.0 to 1.0 and represent the
        proportion of validation dataset. If int, represent the
        absolute number of training dataset

    shuffle: boolean (default=False)
        Whether or not to shuffle the data before splitting
    """
    n_samples = x_train.shape[0]
    if n_va < 1:
        n_va = int(n_samples * n_va)  # from proportion to absolute number
    else:
        n_va = n_va

    if shuffle:
        ind_all = np.random.permutation(n_samples)  # shuffle index
    else:
        ind_all = np.arange(n_samples)  # not shuffle index

    yield ind_all[n_va:], ind_all[:n_va]


def kfold_split(x_train, y_train, n_fold, shuffle=False):
    n_samples = x_train.shape[0]
    # shuffle
    if shuffle:
        ind_all = np.random.permutation(n_samples)
    else:
        ind_all = np.arange(n_samples)
```

```
60
61      f_sizes = [(n_samples + i) // n_fold for i in range(n_fold)]
62      end = 0
63      for f_size in f_sizes:
64          start, end = end, end + f_size
65          ind_tr = np.concatenate([ind_all[end:], ind_all[:start]], axis=0)
66          ind_va = ind_all[start:end]
67          yield ind_tr, ind_va
68
69
70  def plot_cm(y_va, y_va_pred, path):
71      cm = confusion_matrix(y_va, y_va_pred, normalize='true')
72      fig, ax = plt.subplots()
73      sns.heatmap(cm, annot=True, ax=ax)
74      ax.set_xlabel('Predicted Label')
75      ax.set_ylabel('True Label')
76      fig.savefig(path)
77      return fig, ax
78
79
80  def plot_accs(accs, data_name, path, xlabel, xlim=None, log_scale=False):
81      accs_df = pd.DataFrame(accs)
82      fig, ax = plt.subplots(figsize=(7, 4))
83      accs_df.plot.line(marker='x', lw=0.5, ax=ax)
84      if xlim:
85          ax.set_xlim(xlim[0], xlim[1])
86      ax.set_ylim(0, 1)
87      ax.set_title(str(path))
88      ax.set_xlabel(xlabel)
89      ax.set_ylabel('Accuracy')
90      if log_scale:
91          ax.set_xscale('log')
92      ax.grid()
93      fig.tight_layout()
94      fig.savefig(path)
95
96
97  class Trainer:
98      def __init__(self, params, x_train, x_test, y_train, verbose):
99          self.params = params
100         self.x_train = x_train
101         self.x_test = x_test
102         self.y_train = y_train
103         self.verbose = verbose
104         self.y_train_pred = np.zeros_like(y_train)
105         self.y_test_pred = pd.DataFrame()
106         self.scores = defaultdict(list)
107         self.scores_mean = {}
108
109     def train(self, x_tr, y_tr):
110         self.model.fit(x_tr, y_tr)
111
112     def predict(self, x_tr, x_va, y_tr, y_va, ind_va, n_fold, i_fold):
113         y_tr_pred = self.model.predict(x_tr)
114         y_va_pred = self.model.predict(x_va)
115         self.y_train_pred[ind_va] = y_va_pred
116         self.y_test_pred[i_fold] = self.model.predict(self.x_test)
117
118         tr_acc = accuracy_score(y_tr, y_tr_pred)
119         va_acc = accuracy_score(y_va, y_va_pred)
```

```
120         self.scores['tr_acc'].append(tr_acc)
121         self.scores['va_acc'].append(va_acc)
122 
123         if self.verbose:
124             logger.info(f'[fold{i_fold}]\ttr:{tr_acc:.5f}\tva:{va_acc:.5f}')
125 
126     def validate(self, split, n_fold=1, n_va=None, n_tr=None, shuffle=False):
127         if split == 'holdout':
128             cv = tr_va_split(self.x_train, self.x_test, n_va,
shuffle=shuffle)
129         elif split == 'kfold':
130             cv = kfold_split(self.x_train, self.x_test, n_fold,
shuffle=shuffle)
131         else:
132             raise Exception(f'invalid variable name of split: {split}')
133 
134         for i_fold, (ind_tr, ind_va) in enumerate(cv):
135             x_tr, x_va = self.x_train[ind_tr[:n_tr]], self.x_train[ind_va]
136             y_tr, y_va = self.y_train[ind_tr[:n_tr]], self.y_train[ind_va]
137 
138             self.model = SVC(**self.params)
139             self.train(x_tr, y_tr)
140             self.predict(x_tr, x_va, y_tr, y_va, ind_va, n_fold, i_fold)
141 
142         for key in self.scores.keys():
143             self.scores_mean[key] = np.mean(self.scores[key])
144 
145         self.y_test_pred = self.y_test_pred.mean(axis=1).ravel()
146 
```