DataLoader.py

```python
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.model_selection import train_test_split
import joblib
import os


def get_values(value):
    return value.values.reshape(-1, 1)


def load_raw():
    train = pd.read_csv('./data/train.csv')
    test = pd.read_csv('./data/test.csv')

    categorical_features = ['COMPONENT_ARBITRARY', 'YEAR']

    train = train.fillna(0)
    test = test.fillna(0)
    additional_test = train[train["Y_LABEL"] == 1]
    train = train[train["Y_LABEL"] == 0]

    all_X = train.drop(['ID', 'Y_LABEL'], axis=1)
    all_y = train['Y_LABEL']

    test = test.drop(['ID'], axis=1)
    additional_test = additional_test.drop(["ID"], axis=1)[test.columns]

    train_X, val_X, train_y, val_y = train_test_split(all_X, all_y, test_size=0.2)

    scaler = StandardScaler()
    for col in train_X.columns:
        if col not in categorical_features:
            train_X[col] = scaler.fit_transform(get_values(train_X[col]))
            val_X[col] = scaler.transform(get_values(val_X[col]))
            if col in test.columns:
                test[col] = scaler.transform(get_values(test[col]))
                additional_test[col] = scaler.transform(get_values(additional_test[col]))
    le = LabelEncoder()
    for col in categorical_features:
        train_X[col] = le.fit_transform(train_X[col])
        val_X[col] = le.transform(val_X[col])
        if col in test.columns:
            test[col] = le.transform(test[col])
            additional_test[col] = le.transform(additional_test[col])

    # test = pd.concat([test, additional_test])
    return train_X, val_X, train_y, val_y, test, additional_test


class CustomDataset:
    def __init__(self, data_X: pd.DataFrame, data_y, distillation=False):
        super(CustomDataset, self).__init__()
        self.data_X = data_X
        self.data_y = data_y
        self.distillation = distillation
        self.test_stage_features = ['COMPONENT_ARBITRARY', 'ANONYMOUS_1', 'YEAR',
                                    'ANONYMOUS_2', 'AG', 'CO', 'CR', 'CU', 'FE', 'H20',
                                    'MN', 'MO', 'NI', 'PQINDEX', 'TI', 'V', 'V40', 'ZN']

    def __len__(self):
        return len(self.data_X)

    def __getitem__(self, index):
        if self.distillation:
            # 지식 증류 학습 시
            teacher_X = self.data_X.iloc[index].values
            student_X = self.data_X[self.test_stage_features].iloc[index].values
            y = self.data_y.values[index]
            return teacher_X, student_X, y
        else:
            if self.data_y is None:
                test_X = self.data_X.iloc[index].values
                return test_X
            else:
                teacher_X = self.data_X.iloc[index].values
                y = self.data_y.values[index]
                return teacher_X, y


class DataLoader:
    def __init__(self, dataset: CustomDataset, batch_size, shuffle=True):
        self.dataset = dataset
        self.batch_size = batch_size
        self.shuffle = shuffle
        self.on_epoch_end()

    def __len__(self):
        return len(self.dataset) // self.batch_size

    def __iter__(self):
        for item in (self[i] for i in range(len(self))):
            yield item

    def __getitem__(self, idx):
        indices = self.indices[int(idx*self.batch_size): int((idx+1) * self.batch_size)]
        batch_input = self.dataset[indices]
        return batch_input

    def on_epoch_end(self):
        self.indices = list(range(len(self.dataset)))
        if self.shuffle:
            np.random.shuffle(self.indices)
```