GENERAL NOMENCLATURE

VARIABLE NAME	DESCRIPTION
SONG_INDEXES	list of songs that a user has listened to.
x	tensor of SONG_INDEXES. x.shape = (1, idim)
xhat	output of the model, probability of each songs to be liked by the user. xhat.shape = (1, Nsongs)
z	latent space vector corresponding to x. z.shape = (1, dim[-1])

MODEL (model)
scripts from 'modules' folder

model = mymodel.load_mod()
(model = mymodel.load_param() #for loading trained model)

xhat, args = model.forward(x)
MLP: args = None
VAE: args = [mu, logvar]
Uses encoder, decoder (reparametrize for VAE).

mu, logvar = model.encoder(x)
MLP: mu = z, logvar = None

z = model.reparametrize(mu, logvar)
Only for VAE

xhat = model.decoder(z)

z = model.latent(x)
MLP: encoder(x)[0]
VAE: encoder + reparametrize

MODEL CLASS (mymodel)
classes.py

mymodel = MODEL()

VARIABLE	INITIAL VALUE	DATA
idim	100	dimension for x
Nsongs	180198	number of unique songs, dimension of xhat
device	'cuda'	device to perform calculations
loss	None	name of the loss (str)
dim	None	list of dimensions for model (list)
mod	None	name of the model (str)
beta	1.	KLD factor
model_path	'models'	path to the model classes
param_path	'models'	path to trained models
name	None	name used for saving data
embdim	None	string of dim variable
model_lib	None	model class
loss_f	-	loss function

No error: error = None
Error ocurred: error = str

In RECOMMENDER(), to change mymodel it has to be done througth the recom:
recom.mymodel.set_params(...)

mymodel.update_name()
Updates the name variable.

mymodel.set_params(...)
Change variables, except from model_lib, embdim and loss_f, and updates name.

params = mymodel.list_params()
Returns dictionary of current variables.

model, error = mymodel.load_mod()
Load model_lib. Model in eval mode.

model, error = mymodel.load_param()
Outputs model with loaded trained params. Model in eval mode.

error = mymodel.load_loss()
Loads loss from LOSS() class to loss_f.

x = mymodel.input(SONG_INDEXES)

LOSS CLASS
classes.py

myloss = LOSS(mymodel)

VARIABLE	INITIAL VALUE	DATA
Nsongs	mymodel.Nsongs	See mymodel.Nsongs
beta	mymodel.beta	See mymodel.beta
BCELoss	torch.nn.BCELoss()	torch BCE Loss

loss, plot_loss = myloss.X_loss(y, ynew, pars)
y: target
ynew: xhat
pars: [mu, logvar] for KLD losses
loss: torch loss for optimizer
plot_loss: list of values of loss (or parts of loss) for plotting

METADATA CLASS
classes.py

mydata = METADATA()

VARIABLE	INITIAL VALUE	DATA
pcounts_name	'opt_pcounts'	name of the playcounts file
pcounts_path	'data'	path to playcounts file
pcounts	'data/opt_pcounts'	adress to playcounts file
metadata_name	'opt_tags'	name of the metadata file
metadata_path	'data'	path to metadata file
metadata	'data/opt_tags'	adress to metadata file
z_data_path	'z_data'	path to z_data files

mydata.set(...)
Change current variables. Performs mydata.update().

mydata.update()
Updates adress names. Attention, it changes current adress by combination of path and name, so if an adress is selected to be different than this combination it will be altered even though the path and name have not benn changed.

Z_DATA CLASS
post.py

myz = Z_DATA(mymodel = None, mydata = None)

VARIABLE	INITIAL VALUE	DATA
mymodel	None	See mymodel
mydata	None	See mydata

error = myz.z_mean(model)
Calculates z_mean and saves [z_mean, class2idx].

error = myz.z_std(model)
Calculates z_std and saves [z_std, class2idx]. Needs to have z_mean first.

error = myz.plot_z_rand(N_std=10, N_classes = None, classes = None, z_user=None, z_tunning=None, z_mean=None, z_std=None, legend=True)
Plots latent space using N_std points (z = mu + std*N(0,I)) for each class.
If N_classes != None: plots N_classes random classes only.
If classes != None: plots specific classes in this list.

error = myz.plot_z_users(z_user=None, z_tunning=None, z_mean=None, class2idx=None, N_users=-1, model=None, legend=True)
Plots latent space using users from postset test dataset (users have already top tag calculated). Plots a total of N_users users.

TRAINING CLASS
post.py

myloss = TRAINING(plot_path=None, mymodel=None)

VARIABLE	INITIAL VALUE	DATA
loss_graph	[[]]	Stores data from loss for plotting
patience_graph	[[]]	Stores data from pateince for plotting
plot_path	None	path to plot file
mymodel	None	See mymodel

myloss.load()
Change plot_path and mymodel.

error = myloss.load_data()
Loads plot data from a trainning.

myloss.add(plot, patience)
Adds points to plot

myloss.restart()
Activate this function when restarting in the trainning.

myloss.save()
Saves plot data [loss_graph, patience_graph].

error = myloss.plot()
Plots data (if not loaded does load_data()).