I have been using DAE on the legacy dataset, but haven't got around to tailor it for the new dataset, for what it is worth, here is the python class that I have been using in Pytorch:

class Denoise_Autoencoder(nn.Module): def **init**(self, in_dimension, embedding_dimension=10): super().**init**() self.encoder = nn.Sequential(nn.Dropout(p=0.1), nn.Linear(in_dimension, 256), nn.BatchNorm1d(256), nn.Hardswish(), nn.Dropout(p=0.1), nn.Linear(256, 128), nn.BatchNorm1d(128), nn.Hardswish(), nn.Dropout(p=0.1), nn.Linear(128, embedding_dimension),) self.decoder = nn.Sequential(nn.BatchNorm1d(embedding_dimension), nn.Hardswish(), nn.Dropout(p=0.1), nn.Linear(embedding_dimension, 128), nn.BatchNorm1d(128), nn.Hardswish(), nn.Dropout(p=0.1), nn.Linear(128, 256), nn.BatchNorm1d(256), nn.Hardswish(), nn.Dropout(p=0.1), nn.Linear(256, in_dimension),) def forward(self, x): embedding = self.encoder(x) decode = self.decoder(embedding) return embedding, decode

from a crude first look, seems the only significant difference from Yirun's approach is their use of Gaussian noise, which I shall try in due time.

Another departure of my own approach is that I extracted the features, concat them with original feature - it was ok with the old dataset, not sure about the new one - and run it via lightgbm on different tree models