

In []:

```
%cd /content/gdrive/MyDrive/CGM
!ls
# !gzip "/content/drive/MyDrive/CGM/dataset/HIGGS_6M.csv.gz" -d "/content/drive/MyDrive/CGM/dataset"
```

In []:

```
!pip install -U fastbook

Successfully uninstalled torch-1.8.0+cu101
Found existing installation: torchvision 0.9.0+cu101
Uninstalling torchvision-0.9.0+cu101:
  Successfully uninstalled torchvision-0.9.0+cu101
Found existing installation: fastai 1.0.61
Uninstalling fastai-1.0.61:
  Successfully uninstalled fastai-1.0.61
Successfully installed fastai-2.2.7 fastbook-0.0.16 fastcore-1.3.19 fastrelease-0.1.11 gh
api-0.1.16 nbdev-1.1.13 sentencepiece-0.1.95 torch-1.7.1 torchvision-0.8.2
```

In []:

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
import fastbook
fastbook.setup_book()
from fastai.metrics import mse
import torch
import torch.nn as nn
import torch.optim as optim
import torch.utils.data
from torch.autograd import Variable
from torch.utils.data import TensorDataset
from torch.utils.data import DataLoader
from fastai import learner
from fastai.data import core
import time
from fastai.callback import schedule
import os
import numpy as np
from scipy import stats
import seaborn as sns
```

In []:

```
df=pd.read_csv("dataset/HIGGS_6M.csv")
```

Preprocessing dataset

- **Standard Scaling**
- **Min Max Scaling**

In []:

```
dataset=df.to_numpy()
X = dataset[:,1:]
Y = dataset[:,0].astype(int)
print(X[0],Y[0])
print(np.shape(X),np.shape(X[0]),np.shape(Y),np.shape(Y[0]))
from sklearn.preprocessing import StandardScaler
```

```

from sklearn.preprocessing import MinMaxScaler

scaler = StandardScaler()
categorical=[0,9,13,17,21]

for index in range(28):
    X[:,index]=scaler.fit_transform(X[:,index].reshape(-1,1)).reshape(-1)

scaler = MinMaxScaler()
for index in range(28):
    X[:,index]=scaler.fit_transform(X[:,index].reshape(-1,1)).reshape(-1)

```

Compiling DAE Model

In []:

```

class AE_4D_300_LeakyReLU(nn.Module):
    def __init__(self, n_features=28,bottle_neck=8):
        super(AE_4D_300_LeakyReLU, self).__init__()
        self.en1 = nn.Linear(n_features, 300)
        self.en2 = nn.Linear(300, 200)
        self.en3 = nn.Linear(200, 100)
        self.en4 = nn.Linear(100,50)
        self.en5 = nn.Linear(50, bottle_neck)
        self.de1 = nn.Linear(bottle_neck, 50)
        self.de2 = nn.Linear(50, 100)
        self.de3 = nn.Linear(100, 200)
        self.de4 = nn.Linear(200,300)
        self.de5 = nn.Linear(300, n_features)
        self.tanh = nn.Tanh()

    def encode(self, x):
        return self.en5(self.tanh(self.en4(self.tanh(self.en3(self.tanh(self.en2(self.tanh(self.en1(x))))))))))

    def decode(self, x):
        return self.de5(self.tanh(self.de4(self.tanh(self.de3(self.tanh(self.de2(self.tanh(self.de1(self.tanh(x))))))))))

    def forward(self, x):
        z = self.encode(x)
        return self.decode(z)

model = AE_4D_300_LeakyReLU()
model.to('cpu')

```

Out[]:

```

AE_4D_300_LeakyReLU(
  (en1): Linear(in_features=28, out_features=300, bias=True)
  (en2): Linear(in_features=300, out_features=200, bias=True)
  (en3): Linear(in_features=200, out_features=100, bias=True)
  (en4): Linear(in_features=100, out_features=50, bias=True)
  (en5): Linear(in_features=50, out_features=8, bias=True)
  (de1): Linear(in_features=8, out_features=50, bias=True)
  (de2): Linear(in_features=50, out_features=100, bias=True)
  (de3): Linear(in_features=100, out_features=200, bias=True)
  (de4): Linear(in_features=200, out_features=300, bias=True)
  (de5): Linear(in_features=300, out_features=28, bias=True)
  (tanh): Tanh()
)

```

Loading DAE Model from Drive

In []:

```

model_inf = AE_4D_300_LeakyReLU()
model_inf.to('cpu')

```

```
model_inf.load_state_dict(torch.load("dae_model/model.pth"))
```

```
Out[ ]:
```

```
<All keys matched successfully>
```

Generating Encoded Input from DAE Model

You can skip this, next cell loads this from Drive

```
In [ ]:
```

```
from tqdm import tqdm

for i in tqdm(range(np.shape(X)[0]//10**6)):
    data = torch.tensor(X[i*(10**6):(i+1)*(10**6)], dtype=torch.float)
    pred = model_inf.encode(data)
    pred = pred.detach().numpy()
    pred = pred.T
    # pred = np.reshape(pred, (np.shape(pred)[0], 1))
    if i is 0:
        x=pred
        # save(str(i)+'.npy', pred)
    else:
        x=np.concatenate((x,pred), axis=-1)
data = torch.tensor(X[(i+1)*(10**6):], dtype=torch.float)
pred = model_inf.encode(data)
pred = pred.detach().numpy()
pred = pred.T
x=np.concatenate((x,pred), axis=-1)
```

```
In [ ]:
```

```
print(np.shape(x))
x=x.T
print(np.shape(x))
np.save("encoded_x_6M.npy", x)
X = x
del x
```

```
(8, 5999999)
(5999999, 8)
```

```
In [ ]:
```

```
X = np.load("encoded_x_6M.npy")
print(np.shape(X), np.shape(Y))

(5999999, 8) (5999999,)
```

Training XGB Classifier on Encoded input

You can skip training last cell loads the model

```
In [ ]:
```

```
import xgboost
from numpy import loadtxt
from xgboost import XGBClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
```

```
In [ ]:
```

```
seed = 7
test_size = 0.08
X_train, X_valid, y_train, y_valid = train_test_split(X, Y, test_size=test_size, random_
state=seed)
```

```
# Might try a different test split also to check overfitting
```

```
In [ ]:
```

```
eval_set = [(X_valid, y_valid)]
model = XGBClassifier(n_estimators=300, learning_rate=0.1, max_depth=5, gamma=0.1, subsample=0.8, colsample_bytree=0.8)
eval_set = [(X_valid, y_valid)]
model.fit(X_train, y_train, eval_metric="auc", eval_set=eval_set, verbose=True, early_stopping_rounds=5)
```

```
[18:41:09] WARNING: /workspace/src/learner.cc:686: Tree method is automatically selected to be 'approx' for faster speed. To use old behavior (exact greedy algorithm on single machine), set tree_method to 'exact'.
```

```
[0] validation_0-auc:0.566642
```

```
Will train until validation_0-auc hasn't improved in 5 rounds.
```

```
[1] validation_0-auc:0.574211
```

```
[2] validation_0-auc:0.584001
```

```
[3] validation_0-auc:0.587706
```

```
[4] validation_0-auc:0.586605
```

```
[5] validation_0-auc:0.59076
```

```
[6] validation_0-auc:0.591432
```

```
[7] validation_0-auc:0.591808
```

```
[8] validation_0-auc:0.592128
```

```
[9] validation_0-auc:0.593737
```

```
[10] validation_0-auc:0.59503
```

```
[11] validation_0-auc:0.595247
```

```
[12] validation_0-auc:0.595731
```

```
[13] validation_0-auc:0.596987
```

```
[14] validation_0-auc:0.599021
```

```
[15] validation_0-auc:0.599298
```

```
[16] validation_0-auc:0.599405
```

```
[17] validation_0-auc:0.599925
```

```
[18] validation_0-auc:0.600095
```

```
[19] validation_0-auc:0.600445
```

```
[20] validation_0-auc:0.600821
```

```
[21] validation_0-auc:0.601451
```

```
[22] validation_0-auc:0.602242
```

```
[23] validation_0-auc:0.603133
```

```
[24] validation_0-auc:0.603468
```

```
[25] validation_0-auc:0.604218
```

```
[26] validation_0-auc:0.604766
```

```
[27] validation_0-auc:0.605245
```

```
[28] validation_0-auc:0.605708
```

```
[29] validation_0-auc:0.606007
```

```
[30] validation_0-auc:0.606491
```

```
[31] validation_0-auc:0.607231
```

```
[32] validation_0-auc:0.607708
```

```
[33] validation_0-auc:0.608399
```

```
[34] validation_0-auc:0.60879
```

```
[35] validation_0-auc:0.609354
```

```
[36] validation_0-auc:0.609785
```

```
[37] validation_0-auc:0.610237
```

```
[38] validation_0-auc:0.610733
```

```
[39] validation_0-auc:0.610944
```

```
[40] validation_0-auc:0.611082
```

```
[41] validation_0-auc:0.612208
```

```
[42] validation_0-auc:0.612396
```

```
[43] validation_0-auc:0.612654
```

```
[44] validation_0-auc:0.612858
```

```
[45] validation_0-auc:0.61312
```

```
[46] validation_0-auc:0.613304
```

```
[47] validation_0-auc:0.613689
```

```
[48] validation_0-auc:0.614082
```

```
[49] validation_0-auc:0.614407
```

```
[50] validation_0-auc:0.614764
```

```
[51] validation_0-auc:0.615224
```

```
[52] validation_0-auc:0.615441
```

```
[53] validation_0-auc:0.615886
```

```
[54] validation_0-auc:0.616352
```

```
[55] validation_0-auc:0.616398
```

```
[56] validation_0-auc:0.616553
```

[57] validation_0-auc:0.616779
[58] validation_0-auc:0.616899
[59] validation_0-auc:0.617028
[60] validation_0-auc:0.617269
[61] validation_0-auc:0.617462
[62] validation_0-auc:0.618214
[63] validation_0-auc:0.61859
[64] validation_0-auc:0.618881
[65] validation_0-auc:0.619027
[66] validation_0-auc:0.61936
[67] validation_0-auc:0.619673
[68] validation_0-auc:0.61988
[69] validation_0-auc:0.620103
[70] validation_0-auc:0.620319
[71] validation_0-auc:0.620468
[72] validation_0-auc:0.620598
[73] validation_0-auc:0.620805
[74] validation_0-auc:0.62102
[75] validation_0-auc:0.621064
[76] validation_0-auc:0.621348
[77] validation_0-auc:0.621415
[78] validation_0-auc:0.621646
[79] validation_0-auc:0.62225
[80] validation_0-auc:0.622437
[81] validation_0-auc:0.622551
[82] validation_0-auc:0.622628
[83] validation_0-auc:0.62283
[84] validation_0-auc:0.623218
[85] validation_0-auc:0.623681
[86] validation_0-auc:0.624013
[87] validation_0-auc:0.624062
[88] validation_0-auc:0.624233
[89] validation_0-auc:0.624409
[90] validation_0-auc:0.62451
[91] validation_0-auc:0.624805
[92] validation_0-auc:0.624968
[93] validation_0-auc:0.625039
[94] validation_0-auc:0.62506
[95] validation_0-auc:0.625216
[96] validation_0-auc:0.625327
[97] validation_0-auc:0.625465
[98] validation_0-auc:0.625678
[99] validation_0-auc:0.625763
[100] validation_0-auc:0.625837
[101] validation_0-auc:0.626049
[102] validation_0-auc:0.626092
[103] validation_0-auc:0.626381
[104] validation_0-auc:0.626419
[105] validation_0-auc:0.626451
[106] validation_0-auc:0.626522
[107] validation_0-auc:0.626747
[108] validation_0-auc:0.626891
[109] validation_0-auc:0.627102
[110] validation_0-auc:0.627239
[111] validation_0-auc:0.627397
[112] validation_0-auc:0.627743
[113] validation_0-auc:0.627835
[114] validation_0-auc:0.627926
[115] validation_0-auc:0.627958
[116] validation_0-auc:0.628076
[117] validation_0-auc:0.628369
[118] validation_0-auc:0.628429
[119] validation_0-auc:0.628549
[120] validation_0-auc:0.628821
[121] validation_0-auc:0.628966
[122] validation_0-auc:0.629176
[123] validation_0-auc:0.629423
[124] validation_0-auc:0.629628
[125] validation_0-auc:0.629687
[126] validation_0-auc:0.629909
[127] validation_0-auc:0.629969
[128] validation_0-auc:0.630234

[129] validation_0-auc:0.630336
[130] validation_0-auc:0.630355
[131] validation_0-auc:0.630415
[132] validation_0-auc:0.630527
[133] validation_0-auc:0.630864
[134] validation_0-auc:0.630975
[135] validation_0-auc:0.630991
[136] validation_0-auc:0.631121
[137] validation_0-auc:0.631307
[138] validation_0-auc:0.63139
[139] validation_0-auc:0.631416
[140] validation_0-auc:0.63157
[141] validation_0-auc:0.631728
[142] validation_0-auc:0.631806
[143] validation_0-auc:0.631862
[144] validation_0-auc:0.631951
[145] validation_0-auc:0.632094
[146] validation_0-auc:0.632127
[147] validation_0-auc:0.632356
[148] validation_0-auc:0.632415
[149] validation_0-auc:0.632489
[150] validation_0-auc:0.632604
[151] validation_0-auc:0.632673
[152] validation_0-auc:0.63273
[153] validation_0-auc:0.632779
[154] validation_0-auc:0.632836
[155] validation_0-auc:0.633024
[156] validation_0-auc:0.633142
[157] validation_0-auc:0.633258
[158] validation_0-auc:0.633351
[159] validation_0-auc:0.633435
[160] validation_0-auc:0.633604
[161] validation_0-auc:0.633745
[162] validation_0-auc:0.633785
[163] validation_0-auc:0.633935
[164] validation_0-auc:0.633997
[165] validation_0-auc:0.63406
[166] validation_0-auc:0.634317
[167] validation_0-auc:0.634463
[168] validation_0-auc:0.634491
[169] validation_0-auc:0.634526
[170] validation_0-auc:0.634688
[171] validation_0-auc:0.634741
[172] validation_0-auc:0.634793
[173] validation_0-auc:0.634841
[174] validation_0-auc:0.634924
[175] validation_0-auc:0.635168
[176] validation_0-auc:0.635328
[177] validation_0-auc:0.63543
[178] validation_0-auc:0.635494
[179] validation_0-auc:0.635504
[180] validation_0-auc:0.635539
[181] validation_0-auc:0.635595
[182] validation_0-auc:0.635707
[183] validation_0-auc:0.635844
[184] validation_0-auc:0.635977
[185] validation_0-auc:0.636053
[186] validation_0-auc:0.636139
[187] validation_0-auc:0.636223
[188] validation_0-auc:0.636334
[189] validation_0-auc:0.636505
[190] validation_0-auc:0.636548
[191] validation_0-auc:0.636605
[192] validation_0-auc:0.63673
[193] validation_0-auc:0.636792
[194] validation_0-auc:0.636805
[195] validation_0-auc:0.636946
[196] validation_0-auc:0.63707
[197] validation_0-auc:0.637181
[198] validation_0-auc:0.637288
[199] validation_0-auc:0.637383
[200] validation_0-auc:0.637402

[201] validation_0-auc:0.637522
[202] validation_0-auc:0.637629
[203] validation_0-auc:0.637784
[204] validation_0-auc:0.637892
[205] validation_0-auc:0.637918
[206] validation_0-auc:0.637986
[207] validation_0-auc:0.63806
[208] validation_0-auc:0.638173
[209] validation_0-auc:0.638258
[210] validation_0-auc:0.638335
[211] validation_0-auc:0.638353
[212] validation_0-auc:0.638434
[213] validation_0-auc:0.638487
[214] validation_0-auc:0.638634
[215] validation_0-auc:0.638665
[216] validation_0-auc:0.638712
[217] validation_0-auc:0.638777
[218] validation_0-auc:0.638873
[219] validation_0-auc:0.638927
[220] validation_0-auc:0.639026
[221] validation_0-auc:0.639107
[222] validation_0-auc:0.639125
[223] validation_0-auc:0.639196
[224] validation_0-auc:0.639241
[225] validation_0-auc:0.639387
[226] validation_0-auc:0.63943
[227] validation_0-auc:0.639528
[228] validation_0-auc:0.639565
[229] validation_0-auc:0.639664
[230] validation_0-auc:0.639726
[231] validation_0-auc:0.63978
[232] validation_0-auc:0.639828
[233] validation_0-auc:0.639845
[234] validation_0-auc:0.639982
[235] validation_0-auc:0.640094
[236] validation_0-auc:0.640187
[237] validation_0-auc:0.640355
[238] validation_0-auc:0.640442
[239] validation_0-auc:0.640523
[240] validation_0-auc:0.640599
[241] validation_0-auc:0.640673
[242] validation_0-auc:0.640774
[243] validation_0-auc:0.640842
[244] validation_0-auc:0.640944
[245] validation_0-auc:0.640988
[246] validation_0-auc:0.641075
[247] validation_0-auc:0.641129
[248] validation_0-auc:0.641203
[249] validation_0-auc:0.641265
[250] validation_0-auc:0.641296
[251] validation_0-auc:0.641387
[252] validation_0-auc:0.64145
[253] validation_0-auc:0.64152
[254] validation_0-auc:0.641546
[255] validation_0-auc:0.641575
[256] validation_0-auc:0.641675
[257] validation_0-auc:0.641732
[258] validation_0-auc:0.641877
[259] validation_0-auc:0.641915
[260] validation_0-auc:0.64196
[261] validation_0-auc:0.642022
[262] validation_0-auc:0.642219
[263] validation_0-auc:0.642303
[264] validation_0-auc:0.642349
[265] validation_0-auc:0.642452
[266] validation_0-auc:0.642514
[267] validation_0-auc:0.642591
[268] validation_0-auc:0.642647
[269] validation_0-auc:0.642786
[270] validation_0-auc:0.642837
[271] validation_0-auc:0.642961
[272] validation_0-auc:0.642974

```
[273] validation_0-auc:0.643043
[274] validation_0-auc:0.643095
[275] validation_0-auc:0.643191
[276] validation_0-auc:0.643245
[277] validation_0-auc:0.643287
[278] validation_0-auc:0.643317
[279] validation_0-auc:0.643368
[280] validation_0-auc:0.643435
[281] validation_0-auc:0.643496
[282] validation_0-auc:0.643539
[283] validation_0-auc:0.643582
[284] validation_0-auc:0.643595
[285] validation_0-auc:0.643659
[286] validation_0-auc:0.643737
[287] validation_0-auc:0.643779
[288] validation_0-auc:0.643862
[289] validation_0-auc:0.643911
[290] validation_0-auc:0.643961
[291] validation_0-auc:0.644024
[292] validation_0-auc:0.644064
[293] validation_0-auc:0.644079
[294] validation_0-auc:0.644121
[295] validation_0-auc:0.644172
[296] validation_0-auc:0.644245
[297] validation_0-auc:0.644311
[298] validation_0-auc:0.644319
[299] validation_0-auc:0.6444
```

Out[]:

```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=0.8, gamma=0.1,
              learning_rate=0.1, max_delta_step=0, max_depth=5,
              min_child_weight=1, missing=None, n_estimators=300, n_jobs=1,
              nthread=None, objective='binary:logistic', random_state=0,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
              silent=None, subsample=0.8, verbosity=1)
```

In []:

```
y_pred = model.predict(X_valid)
predictions = [round(value) for value in y_pred]
accuracy = accuracy_score(y_valid, predictions)
print("Accuracy: %.2f%%" % (accuracy * 100.0))
```

Accuracy: 60.74%

In []:

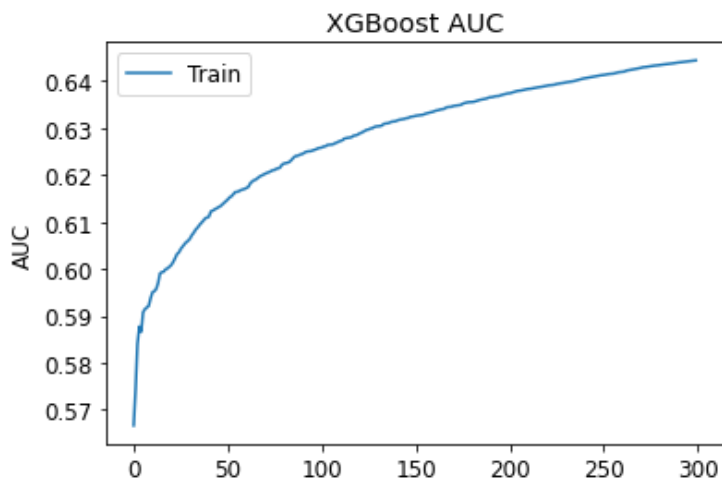
```
from matplotlib import pyplot

results = model.evals_result()
print(results)
epochs = len(results['validation_0']['auc'])
x_axis = range(0, epochs)
# plot log loss
fig, ax = plt.subplots()
ax.plot(x_axis, results['validation_0']['auc'], label='Train')
ax.legend()
pyplot.ylabel('AUC')
pyplot.title('XGBoost AUC')
pyplot.show()
```

```
{'validation_0': {'auc': [0.566642, 0.574211, 0.584001, 0.587706, 0.586605, 0.59076, 0.59
1432, 0.591808, 0.592128, 0.593737, 0.59503, 0.595247, 0.595731, 0.596987, 0.599021, 0.59
9298, 0.599405, 0.599925, 0.600095, 0.600445, 0.600821, 0.601451, 0.602242, 0.603133, 0.6
03468, 0.604218, 0.604766, 0.605245, 0.605708, 0.606007, 0.606491, 0.607231, 0.607708, 0.
608399, 0.60879, 0.609354, 0.609785, 0.610237, 0.610733, 0.610944, 0.611082, 0.612208, 0.
612396, 0.612654, 0.612858, 0.61312, 0.613304, 0.613689, 0.614082, 0.614407, 0.614764, 0.
615224, 0.615441, 0.615886, 0.616352, 0.616398, 0.616553, 0.616779, 0.616899, 0.617028, 0.
617269, 0.617462, 0.618214, 0.61859, 0.618881, 0.619027, 0.61936, 0.619673, 0.61988, 0.6
20103, 0.620319, 0.620468, 0.620598, 0.620805, 0.62102, 0.621064, 0.621348, 0.621415, 0.6
21616, 0.62225, 0.622137, 0.622551, 0.622628, 0.62283, 0.623218, 0.623681, 0.624013, 0.62
4345, 0.624677, 0.625009, 0.625341, 0.625673, 0.626005, 0.626337, 0.626669, 0.627001, 0.627333, 0.627665, 0.62800, 0.628332, 0.628664, 0.62900, 0.629332, 0.629664, 0.63000, 0.630332, 0.630664, 0.63100, 0.631332, 0.631664, 0.63200, 0.632332, 0.632664, 0.63300, 0.633332, 0.633664, 0.63400, 0.634332, 0.634664, 0.63500, 0.635332, 0.635664, 0.63600, 0.636332, 0.636664, 0.63700, 0.637332, 0.637664, 0.63800, 0.638332, 0.638664, 0.63900, 0.639332, 0.639664, 0.64000, 0.640332, 0.640664, 0.64100, 0.641332, 0.641664, 0.64200, 0.642332, 0.642664, 0.64300, 0.643332, 0.643664, 0.64400, 0.644332, 0.644664, 0.64500, 0.645332, 0.645664, 0.64600, 0.646332, 0.646664, 0.64700, 0.647332, 0.647664, 0.64800, 0.648332, 0.648664, 0.64900, 0.649332, 0.649664, 0.65000, 0.650332, 0.650664, 0.65100, 0.651332, 0.651664, 0.65200, 0.652332, 0.652664, 0.65300, 0.653332, 0.653664, 0.65400, 0.654332, 0.654664, 0.65500, 0.655332, 0.655664, 0.65600, 0.656332, 0.656664, 0.65700, 0.657332, 0.657664, 0.65800, 0.658332, 0.658664, 0.65900, 0.659332, 0.659664, 0.66000, 0.660332, 0.660664, 0.66100, 0.661332, 0.661664, 0.66200, 0.662332, 0.662664, 0.66300, 0.663332, 0.663664, 0.66400, 0.664332, 0.664664, 0.66500, 0.665332, 0.665664, 0.66600, 0.666332, 0.666664, 0.66700, 0.667332, 0.667664, 0.66800, 0.668332, 0.668664, 0.66900, 0.669332, 0.669664, 0.67000, 0.670332, 0.670664, 0.67100, 0.671332, 0.671664, 0.67200, 0.672332, 0.672664, 0.67300, 0.673332, 0.673664, 0.67400, 0.674332, 0.674664, 0.67500, 0.675332, 0.675664, 0.67600, 0.676332, 0.676664, 0.67700, 0.677332, 0.677664, 0.67800, 0.678332, 0.678664, 0.67900, 0.679332, 0.679664, 0.68000, 0.680332, 0.680664, 0.68100, 0.681332, 0.681664, 0.68200, 0.682332, 0.682664, 0.68300, 0.683332, 0.683664, 0.68400, 0.684332, 0.684664, 0.68500, 0.685332, 0.685664, 0.68600, 0.686332, 0.686664, 0.68700, 0.687332, 0.687664, 0.68800, 0.688332, 0.688664, 0.68900, 0.689332, 0.689664, 0.69000, 0.690332, 0.690664, 0.69100, 0.691332, 0.691664, 0.69200, 0.692332, 0.692664, 0.69300, 0.693332, 0.693664, 0.69400, 0.694332, 0.694664, 0.69500, 0.695332, 0.695664, 0.69600, 0.696332, 0.696664, 0.69700, 0.697332, 0.697664, 0.69800, 0.698332, 0.698664, 0.69900, 0.699332, 0.699664, 0.70000, 0.700332, 0.700664, 0.70100, 0.701332, 0.701664, 0.70200, 0.702332, 0.702664, 0.70300, 0.703332, 0.703664, 0.70400, 0.704332, 0.704664, 0.70500, 0.705332, 0.705664, 0.70600, 0.706332, 0.706664, 0.70700, 0.707332, 0.707664, 0.70800, 0.708332, 0.708664, 0.70900, 0.709332, 0.709664, 0.71000, 0.710332, 0.710664, 0.71100, 0.711332, 0.711664, 0.71200, 0.712332, 0.712664, 0.71300, 0.713332, 0.713664, 0.71400, 0.714332, 0.714664, 0.71500, 0.715332, 0.715664, 0.71600, 0.716332, 0.716664, 0.71700, 0.717332, 0.717664, 0.71800, 0.718332, 0.718664, 0.71900, 0.719332, 0.719664, 0.72000, 0.720332, 0.720664, 0.72100, 0.721332, 0.721664, 0.72200, 0.722332, 0.722664, 0.72300, 0.723332, 0.723664, 0.72400, 0.724332, 0.724664, 0.72500, 0.725332, 0.725664, 0.72600, 0.726332, 0.726664, 0.72700, 0.727332, 0.727664, 0.72800, 0.728332, 0.728664, 0.72900, 0.729332, 0.729664, 0.73000, 0.730332, 0.730664, 0.73100, 0.731332, 0.731664, 0.73200, 0.732332, 0.732664, 0.73300, 0.733332, 0.733664, 0.73400, 0.734332, 0.734664, 0.73500, 0.735332, 0.735664, 0.73600, 0.736332, 0.736664, 0.73700, 0.737332, 0.737664, 0.73800, 0.738332, 0.738664, 0.73900, 0.739332, 0.739664, 0.74000, 0.740332, 0.740664, 0.74100, 0.741332, 0.741664, 0.74200, 0.742332, 0.742664, 0.74300, 0.743332, 0.743664, 0.74400, 0.744332, 0.744664, 0.74500, 0.745332, 0.745664, 0.74600, 0.746332, 0.746664, 0.74700, 0.747332, 0.747664, 0.74800, 0.748332, 0.748664, 0.74900, 0.749332, 0.749664, 0.75000, 0.750332, 0.750664, 0.75100, 0.751332, 0.751664, 0.75200, 0.752332, 0.752664, 0.75300, 0.753332, 0.753664, 0.75400, 0.754332, 0.754664, 0.75500, 0.755332, 0.755664, 0.75600, 0.756332, 0.756664, 0.75700, 0.757332, 0.757664, 0.75800, 0.758332, 0.758664, 0.75900, 0.759332, 0.759664, 0.76000, 0.760332, 0.760664, 0.76100, 0.761332, 0.761664, 0.76200, 0.762332, 0.762664, 0.76300, 0.763332, 0.763664, 0.76400, 0.764332, 0.764664, 0.76500, 0.765332, 0.765664, 0.76600, 0.766332, 0.766664, 0.76700, 0.767332, 0.767664, 0.76800, 0.768332, 0.768664, 0.76900, 0.769332, 0.769664, 0.77000, 0.770332, 0.770664, 0.77100, 0.771332, 0.771664, 0.77200, 0.772332, 0.772664, 0.77300, 0.773332, 0.773664, 0.77400, 0.774332, 0.774664, 0.77500, 0.775332, 0.775664, 0.77600, 0.776332, 0.776664, 0.77700, 0.777332, 0.777664, 0.77800, 0.778332, 0.778664, 0.77900, 0.779332, 0.779664, 0.78000, 0.780332, 0.780664, 0.78100, 0.781332, 0.781664, 0.78200, 0.782332, 0.782664, 0.78300, 0.783332, 0.783664, 0.78400, 0.784332, 0.784664, 0.78500, 0.785332, 0.785664, 0.78600, 0.786332, 0.786664, 0.78700, 0.787332, 0.787664, 0.78800, 0.788332, 0.788664, 0.78900, 0.789332, 0.789664, 0.79000, 0.790332, 0.790664, 0.79100, 0.791332, 0.791664, 0.79200, 0.792332, 0.792664, 0.79300, 0.793332, 0.793664, 0.79400, 0.794332, 0.794664, 0.79500, 0.795332, 0.795664, 0.79600, 0.796332, 0.796664, 0.79700, 0.797332, 0.797664, 0.79800, 0.798332, 0.798664, 0.79900, 0.799332, 0.799664, 0.80000, 0.800332, 0.800664, 0.80100, 0.801332, 0.801664, 0.80200, 0.802332, 0.802664, 0.80300, 0.803332, 0.803664, 0.80400, 0.804332, 0.804664, 0.80500, 0.805332, 0.805664, 0.80600, 0.806332, 0.806664, 0.80700, 0.807332, 0.807664, 0.80800, 0.808332, 0.808664, 0.80900, 0.809332, 0.809664, 0.81000, 0.810332, 0.810664, 0.81100, 0.811332, 0.811664, 0.81200, 0.812332, 0.812664, 0.81300, 0.813332, 0.813664, 0.81400, 0.814332, 0.814664, 0.81500, 0.815332, 0.815664, 0.81600, 0.816332, 0.816664, 0.81700, 0.817332, 0.817664, 0.81800, 0.818332, 0.818664, 0.81900, 0.819332, 0.819664, 0.82000, 0.820332, 0.820664, 0.82100, 0.821332, 0.821664, 0.82200, 0.822332, 0.822664, 0.82300, 0.823332, 0.823664, 0.82400, 0.824332, 0.824664, 0.82500, 0.825332, 0.825664, 0.82600, 0.826332, 0.826664, 0.82700, 0.827332, 0.827664, 0.82800, 0.828332, 0.828664, 0.82900, 0.829332, 0.829664, 0.83000, 0.830332, 0.830664, 0.83100, 0.831332, 0.831664, 0.83200, 0.832332, 0.832664, 0.83300, 0.833332, 0.833664, 0.83400, 0.834332, 0.834664, 0.83500, 0.835332, 0.835664, 0.83600, 0.836332, 0.836664, 0.83700, 0.837332, 0.837664, 0.83800, 0.838332, 0.838664, 0.83900, 0.839332, 0.839664, 0.84000, 0.840332, 0.840664, 0.84100, 0.841332, 0.841664, 0.84200, 0.842332, 0.842664, 0.84300, 0.843332, 0.843664, 0.84400, 0.844332, 0.844664, 0.84500, 0.845332, 0.845664, 0.84600, 0.846332, 0.846664, 0.84700, 0.847332, 0.847664, 0.84800, 0.848332, 0.848664, 0.84900, 0.849332, 0.849664, 0.85000, 0.850332, 0.850664, 0.85100, 0.851332, 0.851664, 0.85200, 0.852332, 0.852664, 0.85300, 0.853332, 0.853664, 0.85400, 0.854332, 0.854664, 0.85500, 0.855332, 0.855664, 0.85600, 0.856332, 0.856664, 0.85700, 0.857332, 0.857664, 0.85800, 0.858332, 0.858664, 0.85900, 0.859332, 0.859664, 0.86000, 0.860332, 0.860664, 0.86100, 0.861332, 0.861664, 0.86200, 0.862332, 0.862664, 0.86300, 0.863332, 0.863664, 0.86400, 0.864332, 0.864664, 0.86500, 0.865332, 0.865664, 0.86600, 0.866332, 0.866664, 0.86700, 0.867332, 0.867664, 0.86800, 0.868332, 0.868664, 0.86900, 0.869332, 0.869664, 0.87000, 0.870332, 0.870664, 0.87100, 0.871332, 0.871664, 0.87200, 0.872332, 0.872664, 0.87300, 0.873332, 0.873664, 0.87400, 0.874332, 0.874664, 0.87500, 0.875332, 0.875664, 0.87600, 0.876332, 0.876664, 0.87700, 0.877332, 0.877664, 0.87800, 0.878332, 0.878664, 0.87900, 0.879332, 0.879664, 0.88000, 0.880332, 0.880664, 0.88100, 0.881332, 0.881664, 0.88200, 0.882332, 0.882664, 0.88300, 0.883332, 0.883664, 0.88400, 0.884332, 0.884664, 0.88500, 0.885332, 0.885664, 0.88600, 0.886332, 0.886664, 0.88700, 0.887332, 0.887664, 0.88800, 0.888332, 0.888664, 0.88900, 0.889332, 0.889664, 0.89000, 0.890332, 0.890664, 0.89100, 0.891332, 0.891664, 0.89200, 0.892332, 0.892664, 0.89300, 0.893332, 0.893664, 0.89400, 0.894332, 0.894664, 0.89500, 0.895332, 0.895664, 0.89600, 0.896332, 0.896664, 0.89700, 0.897332, 0.897664, 0.89800, 0.898332, 0.898664, 0.89900, 0.899332, 0.899664, 0.90000, 0.900332, 0.900664, 0.90100, 0.901332, 0.901664, 0.90200, 0.902332, 0.902664, 0.90300, 0.903332, 0.903664, 0.90400, 0.904332, 0.904664, 0.90500, 0.905332, 0.905664, 0.90600, 0.906332, 0.906664, 0.90700, 0.907332, 0.907664, 0.90800, 0.908332, 0.908664, 0.90900, 0.909332, 0.909664, 0.91000, 0.910332, 0.910664, 0.91100, 0.911332, 0.911664, 0.91200, 0.912332, 0.912664, 0.91300, 0.913332, 0.913664, 0.91400, 0.914332, 0.914664, 0.91500, 0.915332, 0.915664, 0.91600, 0.916332, 0.916664, 0.91700, 0.917332, 0.917664, 0.91800, 0.918332, 0.918664, 0.91900, 0.919332, 0.919664, 0.92000, 0.920332, 0.920664, 0.92100, 0.921332, 0.921664, 0.92200, 0.922332, 0.922664, 0.92300, 0.923332, 0.923664, 0.92400, 0.924332, 0.924664, 0.92500, 0.925332, 0.925664, 0.92600, 0.926332, 0.926664, 0.92700, 0.927332, 0.927664, 0.92800, 0.928332, 0.928664, 0.92900, 0.929332, 0.929664, 0.93000, 0.930332, 0.930664, 0.93100, 0.931332, 0.931664, 0.93200, 0.932332, 0.932664, 0.93300, 0.933332, 0.933664, 0.93400, 0.934332, 0.934664, 0.93500, 0.935332, 0.935664, 0.93600, 0.936332, 0.936664, 0.93700, 0.937332, 0.937664, 0.93800, 0.938332, 0.938664, 0.93900, 0.939332, 0.939664, 0.94000, 0.940332, 0.940664, 0.94100, 0.941332, 0.941664, 0.94200, 0.942332, 0.942664, 0.94300, 0.943332, 0.943664, 0.94400, 0.944332, 0.944664, 0.94500, 0.945332, 0.945664, 0.94600, 0.946332, 0.946664, 0.94700, 0.947332, 0.947664, 0.94800, 0.948332, 0.948664, 0.94900, 0.949332, 0.949664, 0.95000, 0.950332, 0.950664, 0.95100, 0.951332, 0.951664, 0.95200, 0.952332, 0.952664, 0.95300, 0.953332, 0.953664, 0.95400, 0.954332, 0.954664, 0.95500, 0.955332, 0.955664, 0.95600, 0.956332, 0.956664, 0.95700, 0.957332, 0.957664, 0.95800, 0.958332, 0.958664, 0.95900, 0.959332, 0.959664, 0.96000, 0.960332, 0.960664, 0.96100, 0.961332, 0.961664, 0.96200, 0.962332, 0.962664, 0.96300, 0.963332, 0.963664, 0.96400, 0.964332, 0.964664, 0.96500, 0.965332, 0.965664, 0.96600, 0.966332, 0.966664, 0.96700, 0.967332, 0.967664, 0.96800, 0.968332, 0.968664, 0.96900, 0.969332, 0.969664, 0.97000, 0.970332, 0.970664, 0.97100, 0.971332, 0.971664, 0.97200, 0.972332, 0.972664, 0.97300, 0.973332, 0.973664, 0.97400, 0.974332, 0.974664, 0.97500, 0.975332, 0.975664, 0.97600, 0.976332, 0.976664, 0.97700, 0.977332, 0.977664, 0.97800, 0.978332, 0.978664, 0.97900, 0.979332, 0.979664, 0.98000, 0.980332, 0.980664, 0.9
```



```
21040, 0.62223, 0.622437, 0.622531, 0.622628, 0.62268, 0.622718, 0.622801, 0.624013, 0.624062, 0.624233, 0.624409, 0.62451, 0.624805, 0.624968, 0.625039, 0.62506, 0.625216, 0.625327, 0.625465, 0.625678, 0.625763, 0.625837, 0.626049, 0.626092, 0.626381, 0.626419, 0.626451, 0.626522, 0.626747, 0.626891, 0.627102, 0.627239, 0.627397, 0.627743, 0.627835, 0.627926, 0.627958, 0.628076, 0.628369, 0.628429, 0.628549, 0.628821, 0.628966, 0.629176, 0.629423, 0.629628, 0.629687, 0.629909, 0.629969, 0.630234, 0.630336, 0.630355, 0.630415, 0.630527, 0.630864, 0.630975, 0.630991, 0.631121, 0.631307, 0.63139, 0.631416, 0.63157, 0.631728, 0.631806, 0.631862, 0.631951, 0.632094, 0.632127, 0.632356, 0.632415, 0.632489, 0.632604, 0.632673, 0.63273, 0.632779, 0.632836, 0.633024, 0.633142, 0.633258, 0.633351, 0.633435, 0.633604, 0.633745, 0.633785, 0.633935, 0.633997, 0.63406, 0.634317, 0.634463, 0.634491, 0.634526, 0.634688, 0.634741, 0.634793, 0.634841, 0.634924, 0.635168, 0.635328, 0.63543, 0.635494, 0.635504, 0.635539, 0.635595, 0.635707, 0.635844, 0.635977, 0.636053, 0.636139, 0.636223, 0.636334, 0.636505, 0.636548, 0.636605, 0.63673, 0.636792, 0.636805, 0.636946, 0.63707, 0.637181, 0.637288, 0.637383, 0.637402, 0.637522, 0.637629, 0.637784, 0.637892, 0.637918, 0.637986, 0.63806, 0.638173, 0.638258, 0.638335, 0.638353, 0.638434, 0.638487, 0.638634, 0.638665, 0.638712, 0.638777, 0.638873, 0.638927, 0.639026, 0.639107, 0.639125, 0.639196, 0.639241, 0.639387, 0.63943, 0.639528, 0.639565, 0.639664, 0.639726, 0.63978, 0.639828, 0.639845, 0.639982, 0.640094, 0.640187, 0.640355, 0.640442, 0.640523, 0.640599, 0.640673, 0.640774, 0.640842, 0.640944, 0.640988, 0.641075, 0.641129, 0.641203, 0.641265, 0.641296, 0.641387, 0.64145, 0.64152, 0.641546, 0.641575, 0.641675, 0.641732, 0.641877, 0.641915, 0.64196, 0.642022, 0.642219, 0.642303, 0.642349, 0.642452, 0.642514, 0.642591, 0.642647, 0.642786, 0.642837, 0.642961, 0.642974, 0.643043, 0.643095, 0.643191, 0.643245, 0.643287, 0.643317, 0.643368, 0.643435, 0.643496, 0.643539, 0.643582, 0.643595, 0.643659, 0.643737, 0.643779, 0.643862, 0.643911, 0.643961, 0.644024, 0.644064, 0.644079, 0.644121, 0.644172, 0.644245, 0.644311, 0.644319, 0.6444]]}
```



```
In [ ]:
```

```
import pickle
pickle.dump(model, open("xgb_encoded.pickle.dat", "wb"))
```

```
In [ ]:
```

```
"""
Checking whether load and save is working okay
"""
loaded_model = pickle.load(open("xgb_encoded.pickle.dat", "rb"))
y_pred = loaded_model.predict(X_valid)
predictions = [round(value) for value in y_pred]
accuracy = accuracy_score(y_valid, predictions)
print("Accuracy: %.2f%%" % (accuracy * 100.0))
```

```
Accuracy: 60.74%
```

Incremental Training for XGB

Not needed anymore, found another fix, read appendix of technical report

```
In [ ]:
```

```
# import xgboost
# from numpy import loadtxt
# from xgboost import XGBClassifier
```

```

# from sklearn.model_selection import train_test_split
# from sklearn.metrics import accuracy_score

# BATCH_SIZE = 10**4
# def trainXGB():
#     model = None

#     for i in range(np.shape(X)[0]//BATCH_SIZE):
#         print("Iteration: "+str(i))

#         x = X[i*BATCH_SIZE:(i+1)*BATCH_SIZE]
#         y = Y[i*BATCH_SIZE:(i+1)*BATCH_SIZE]

#         x = model_inf(torch.tensor(x, dtype=torch.float)).detach().numpy()
#         y = y

#         seed = 7
#         test_size = 0.08
#         X_train, X_valid, y_train, y_valid = train_test_split(x, y, test_size=test_size, random_state=seed)

#         eval_set = [(X_valid, y_valid)]
#         model = xgboost.train({
#             'update': 'refresh',
#             'process_type': 'update',
#             'refresh_leaf': True,
#             'silent': False,
#             'obj': 'auc',
#             }, dtrain=xgboost.DMatrix(X_train, y_train), xgb_model=model)
#         y_pred = model.predict(xgboost.DMatrix(X_valid))
#         predictions = [round(value) for value in y_pred]
#         accuracy = accuracy_score(y_valid, predictions)
#         print("Accuracy: %.2f%%" % (accuracy * 100.0))

#     return model

```

In []:

```
# trained_model = trainXGB()
```