Open in Colab

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 ghapi-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 [ ]:

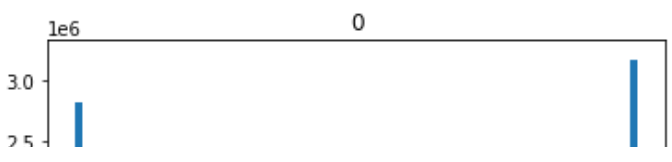
```
df=pd.read_csv("dataset/HIGGS_6M.csv")
```

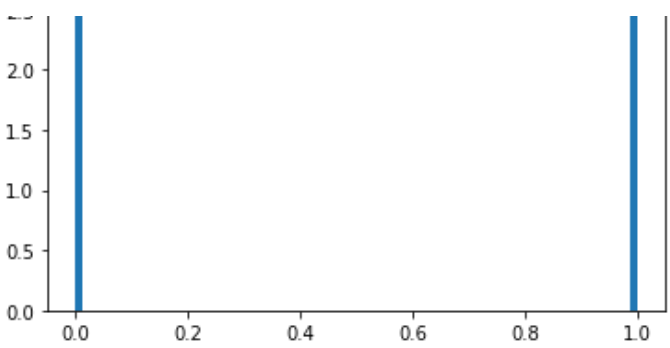
In [ ]:

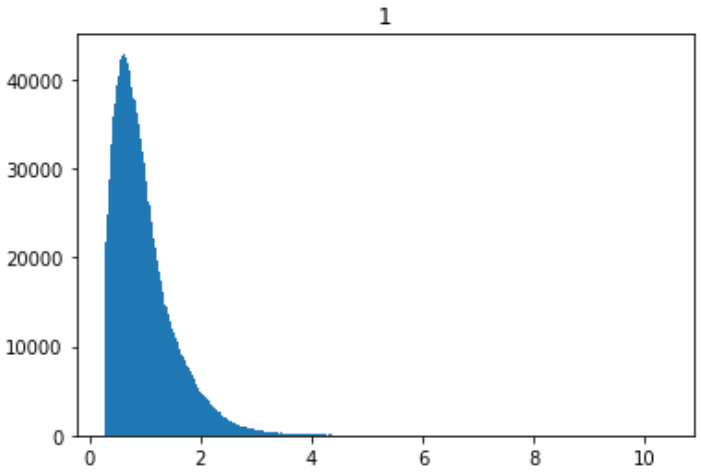
```
dataset=df.to_numpy()
```

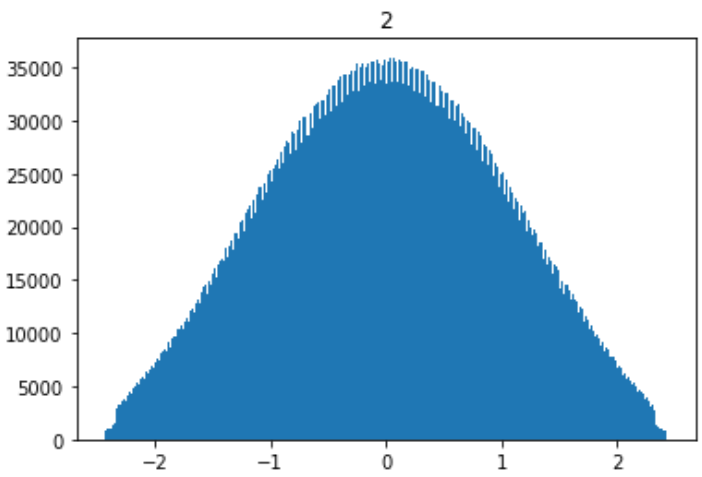
In [ ]:

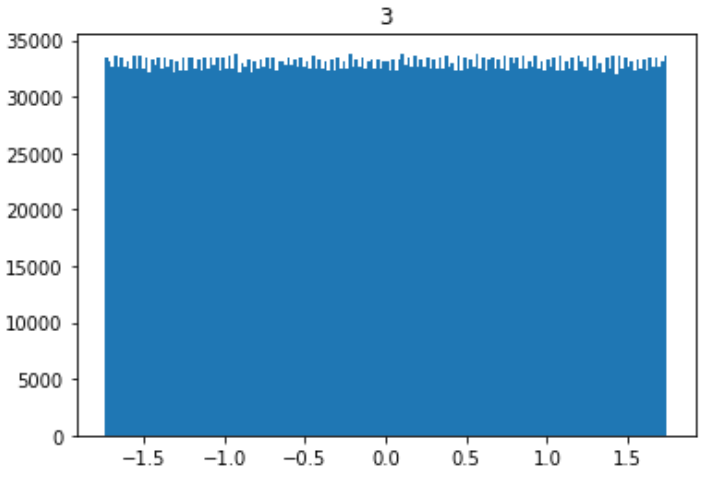
```
for i in range(29):
  plt.hist(dataset[:,i],bins='auto')
  plt.title(str(i))
  plt.show()
```

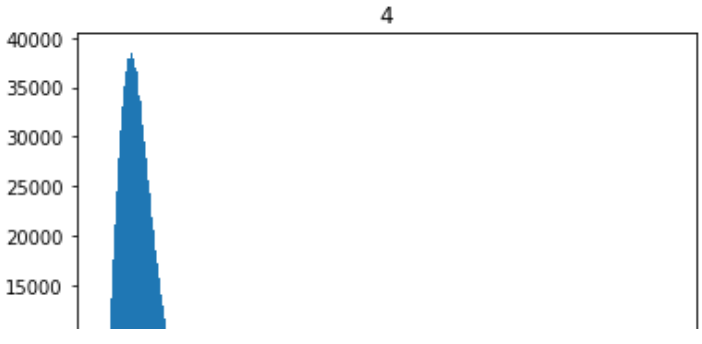


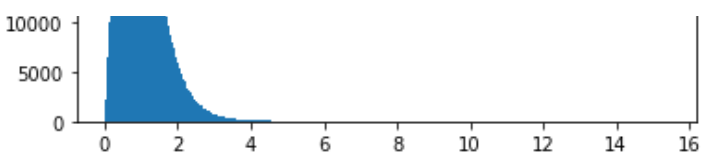

2.0
1.5
1.0
0.5
0.0
0.0
0.2
0.4
0.6
0.8
1.0


1
40000
30000
20000
10000
0
0
2
4
6
8
10

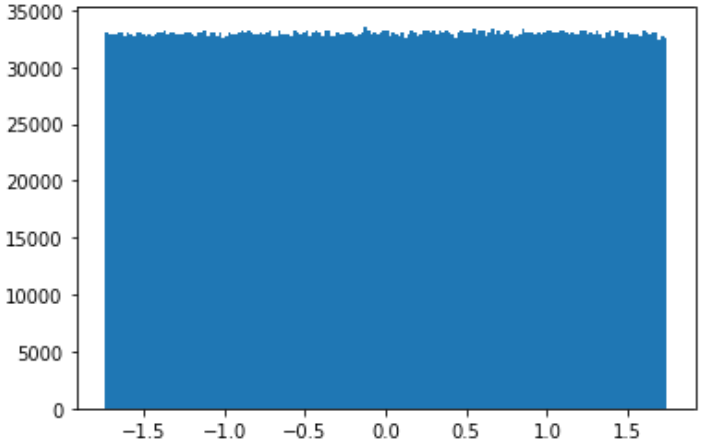

2
35000
30000
25000
20000
15000
10000
5000
0
−2
−1
0
1
2


3
35000
30000
25000
20000
15000
10000
5000
0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5

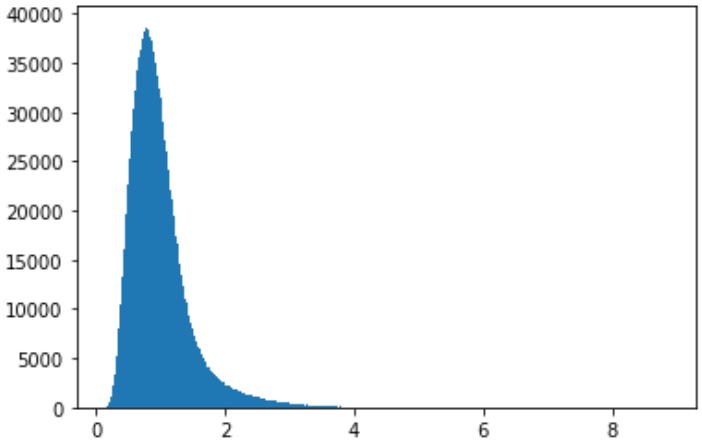

4
40000
35000
30000
25000
20000
15000


10000
5000
0
0
2
4
6
8
10
12
14
16

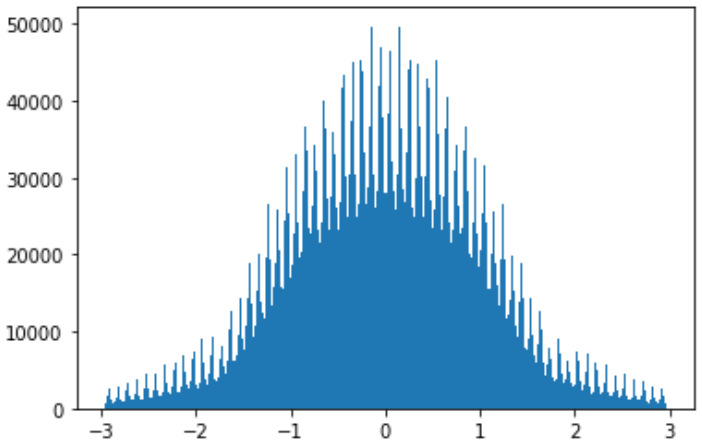
5


35000
30000
25000
20000
15000
10000
5000
0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5

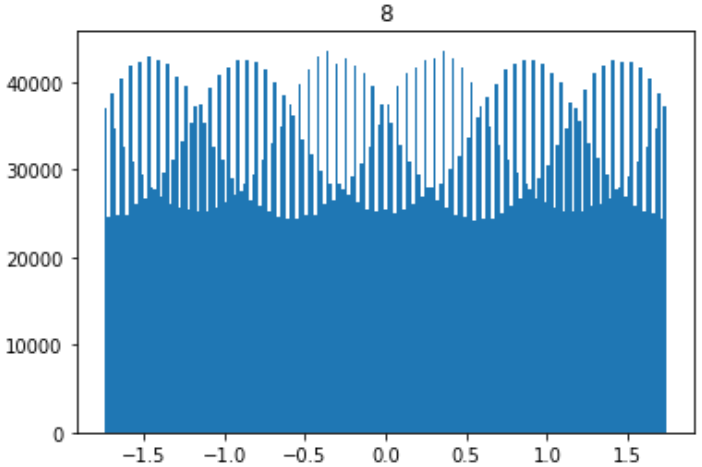
6

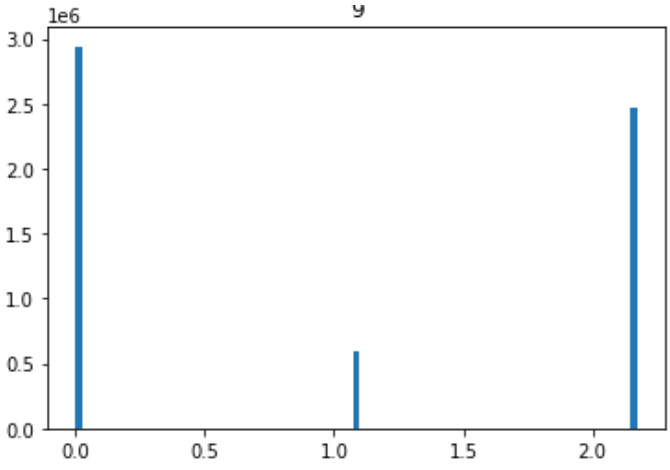

40000
35000
30000
25000
20000
15000
10000
5000
0
0
2
4
6
8

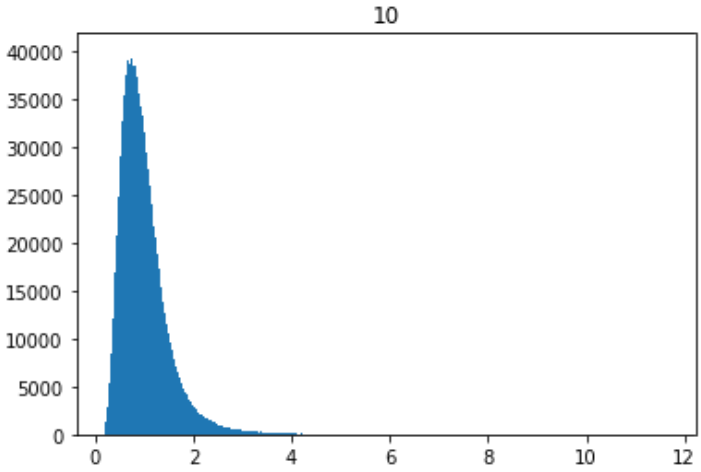
7

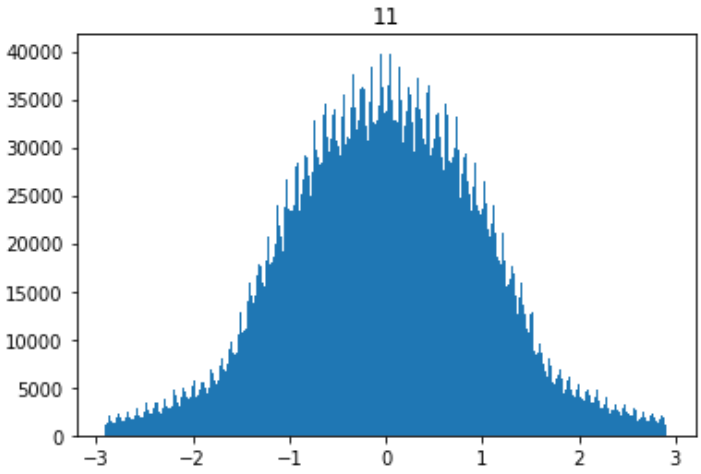

50000
40000
30000
20000
10000
0
−3
−2
−1
0
1
2
3

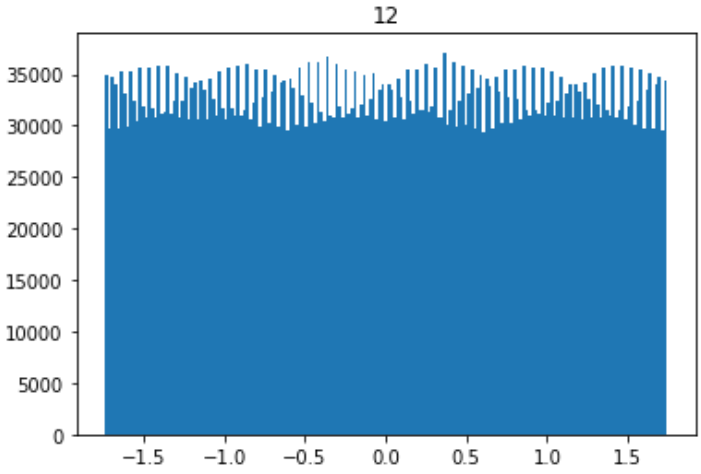
8

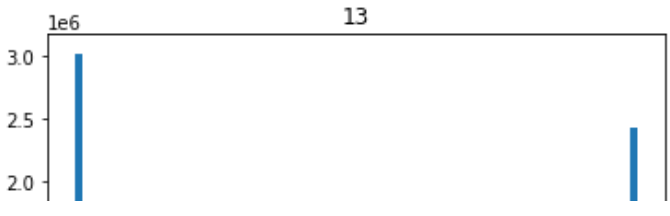

40000
30000
20000
10000
0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5

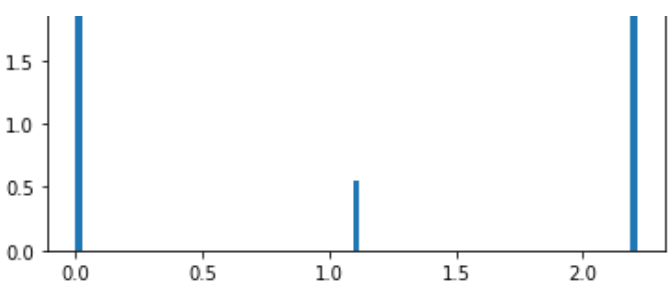

9
1e6
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0.0
0.5
1.0
1.5
2.0

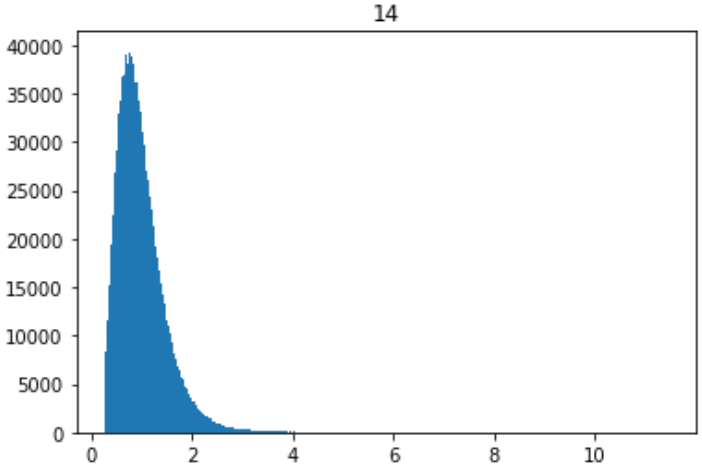

10
40000
35000
30000
25000
20000
15000
10000
5000
0
0
2
4
6
8
10
12

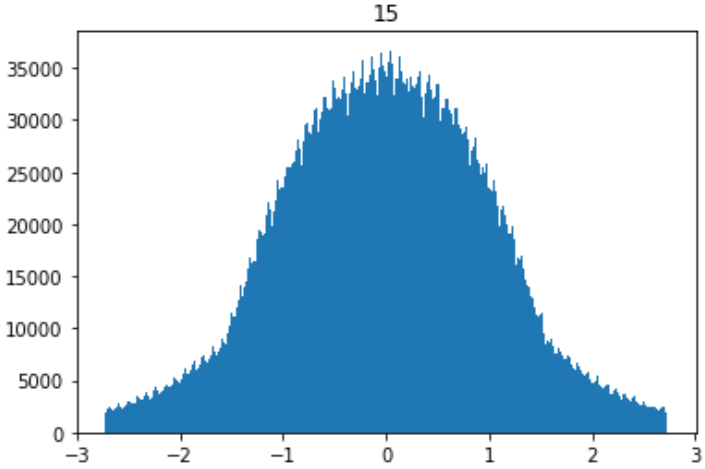

11
40000
35000
30000
25000
20000
15000
10000
5000
0
−3
−2
−1
0
1
2
3

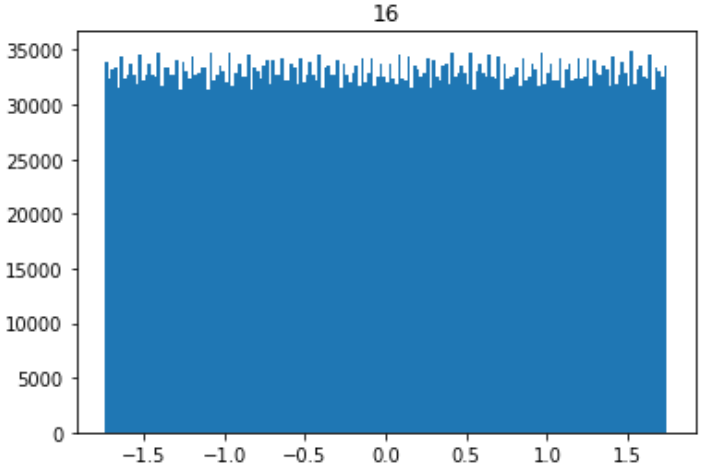

12
35000
30000
25000
20000
15000
10000
5000
0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5


13
1e6
3.0
2.5
2.0

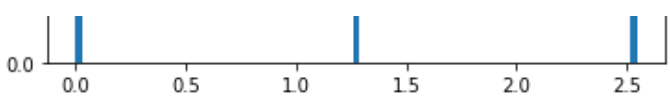

1.5
1.0
0.5
0.0
0.0
0.5
1.0
1.5
2.0

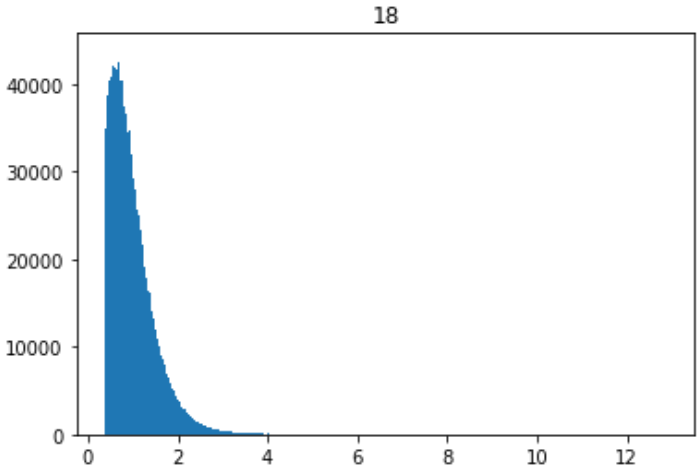

14
40000
35000
30000
25000
20000
15000
10000
5000
0
0
2
4
6
8
10

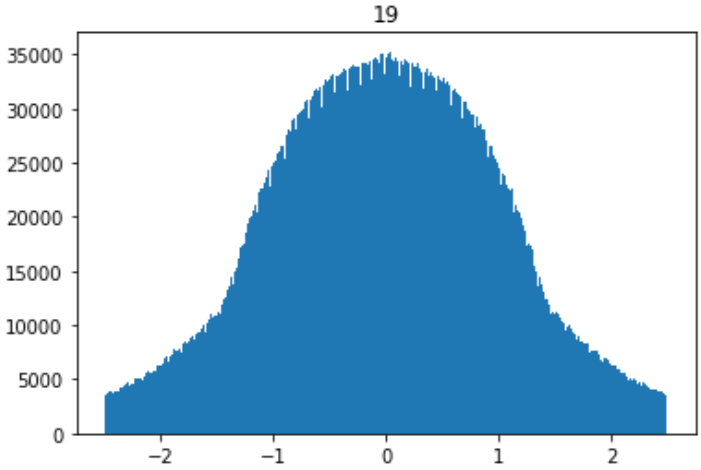

15
35000
30000
25000
20000
15000
10000
5000
0
−3
−2
−1
0
1
2
3

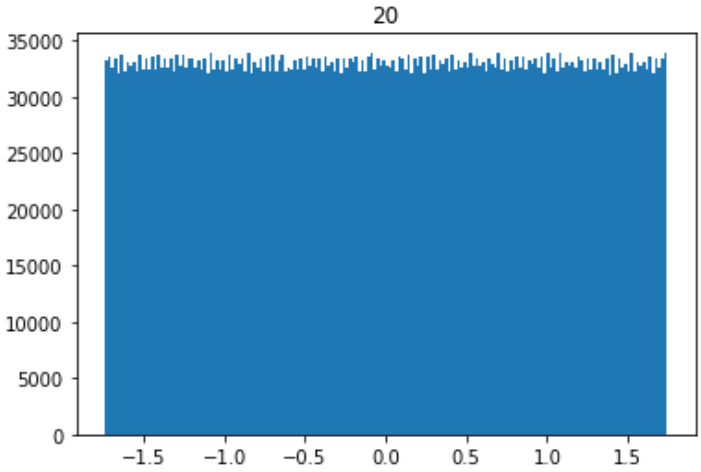

16
35000
30000
25000
20000
15000
10000
5000
0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5

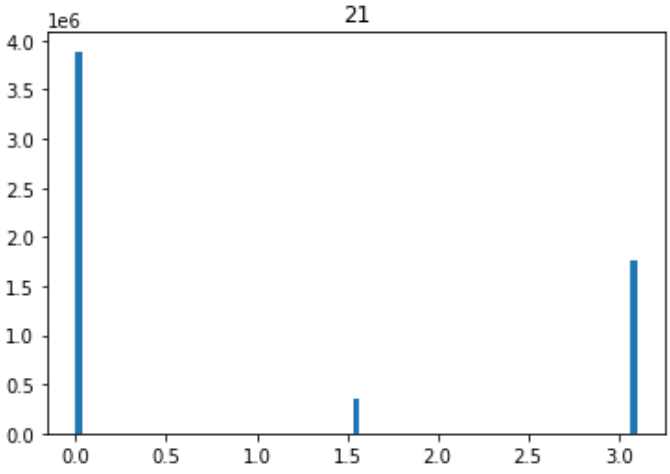

17
1e6
3.5
3.0
2.5
2.0
1.5
1.0
0.5

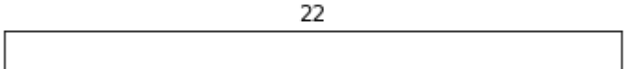

0.0
0.0
0.5
1.0
1.5
2.0
2.5

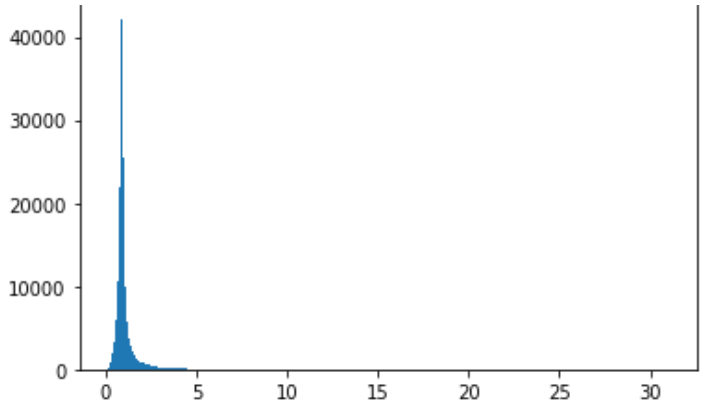

18
40000
30000
20000
10000
0
0
2
4
6
8
10
12

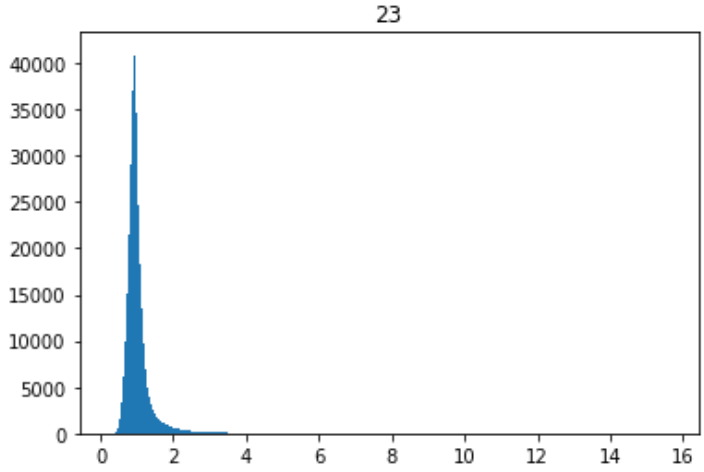

19
35000
30000
25000
20000
15000
10000
5000
0
−2
−1
0
1
2

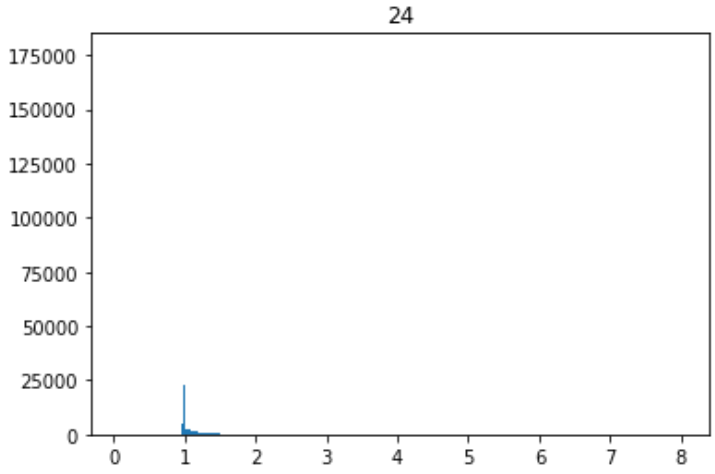

20
35000
30000
25000
20000
15000
10000
5000
0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5

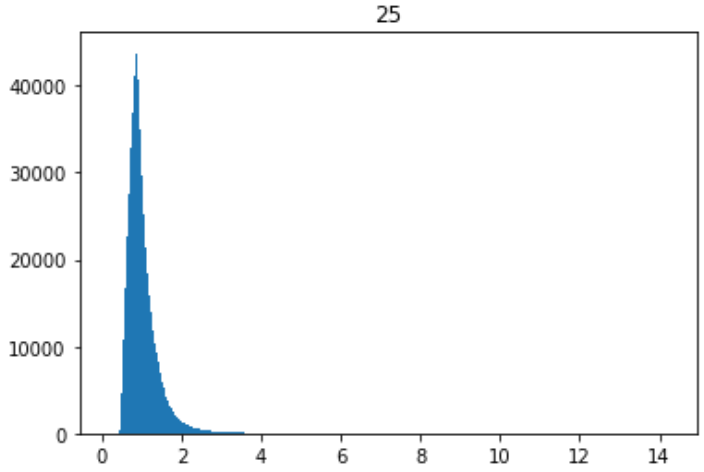

21
1e6
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0.0
0.5
1.0
1.5
2.0
2.5
3.0

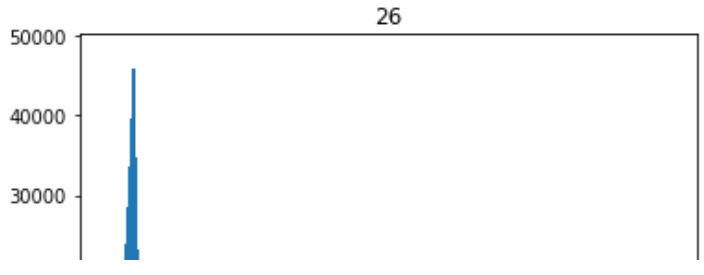

22

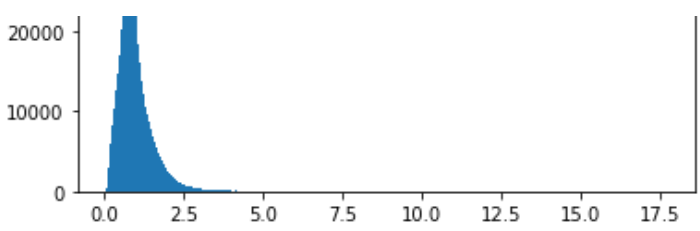

40000
30000
20000
10000
0
0
5
10
15
20
25
30

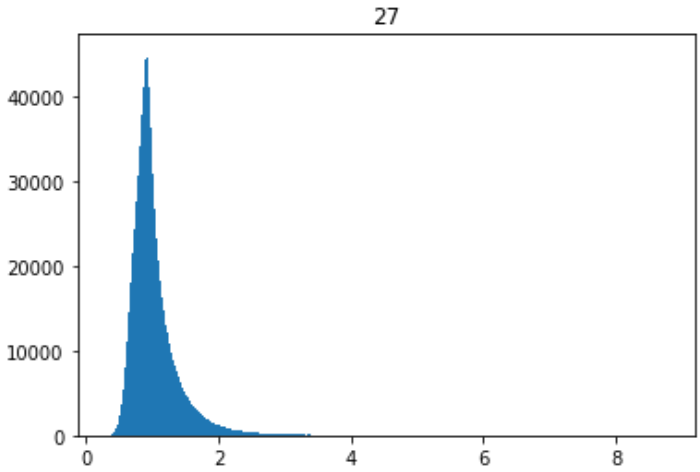

23
40000
35000
30000
25000
20000
15000
10000
5000
0
0
2
4
6
8
10
12
14
16

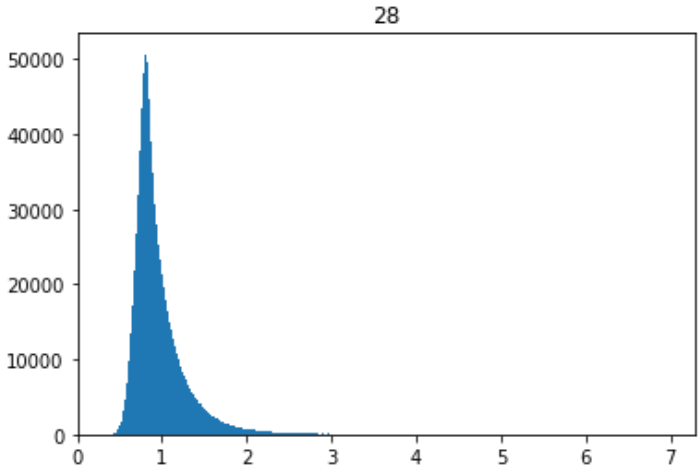

24
175000
150000
125000
100000
75000
50000
25000
0
0
1
2
3
4
5
6
7
8


25
40000
30000
20000
10000
0
0
2
4
6
8
10
12
14


26
50000
40000
30000







Preprocessing

- **Log Transform**
- **Standard Scaling**
- **Min Max Scaling**
- **Rounding categorical features to integers**

You can skip training, jump to the last cell to load from drive

In [ ]:

```
mod_dataset=dataset
logtransform=[1,4,6,10,14,18,22,23,24,25,26,27,28]
for index in logtransform:
  mod_dataset[:,index]=np.log(mod_dataset[:,index])
```

In [ ]:

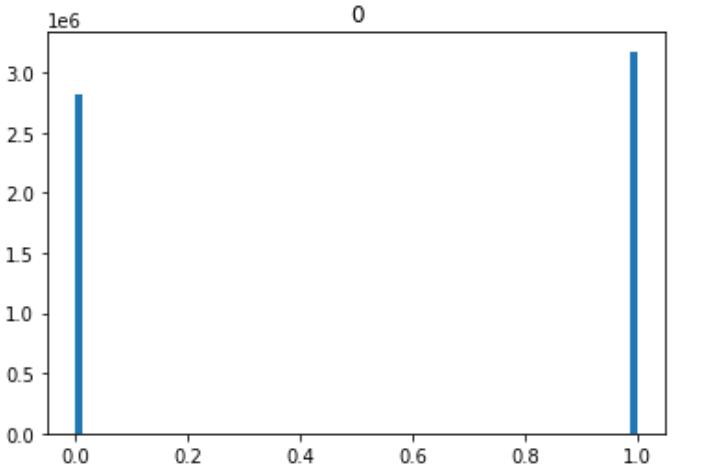
```
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler

scaler = StandardScaler()
categorical=[0,9,13,17,21]
non_categorical=[]
for i in range(29):
  if i not in categorical:
    non_categorical.append(i)
print(non_categorical)
```

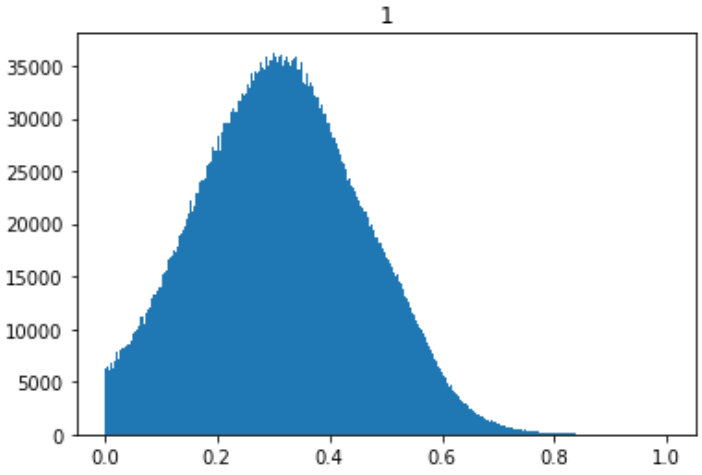
```
for index in non_categorical:
  mod_dataset[:,index]=scaler.fit_transform(mod_dataset[:,index].reshape(-1,1)).reshape(
-1)

scaler = MinMaxScaler()
for index in non_categorical:
  mod_dataset[:,index]=scaler.fit_transform(mod_dataset[:,index].reshape(-1,1)).reshape(
-1)
```

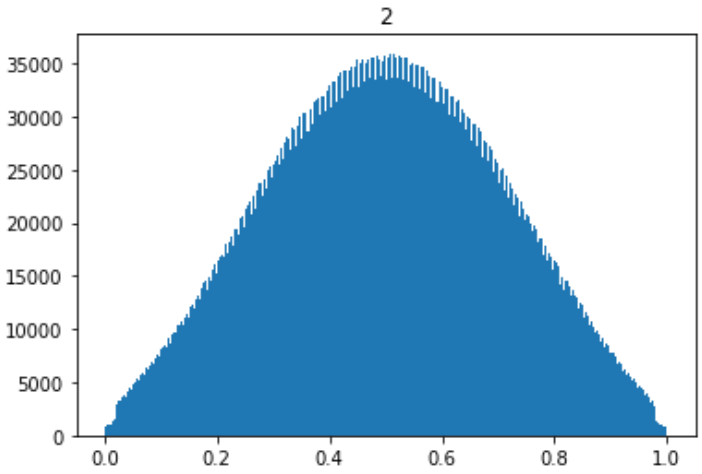
```
[1, 2, 3, 4, 5, 6, 7, 8, 10, 11, 12, 14, 15, 16, 18, 19, 20, 22, 23, 24, 25, 26, 27, 28]
```

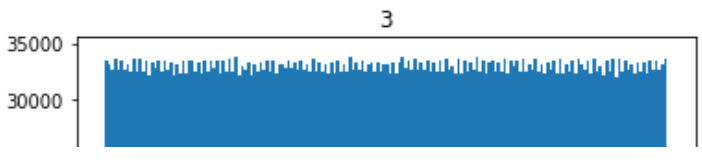
In [ ]:

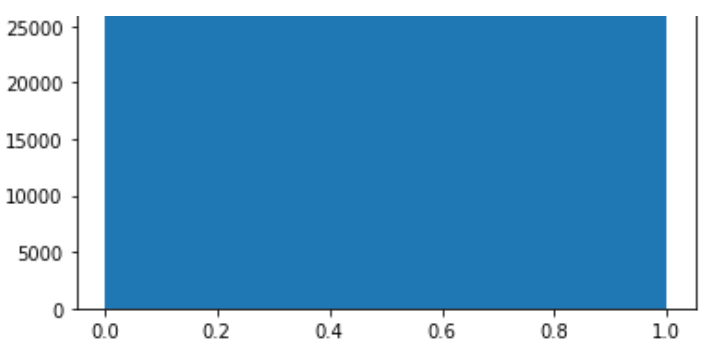
```
for i in range(29):
  plt.hist(mod_dataset[:,i],bins='auto')
  plt.title(str(i))
  plt.show()
```



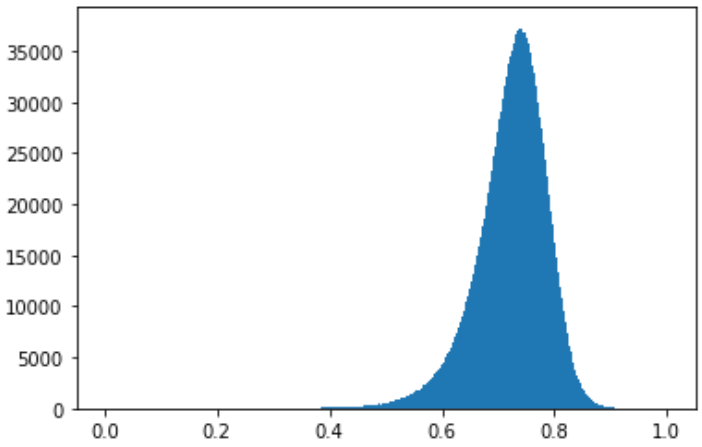




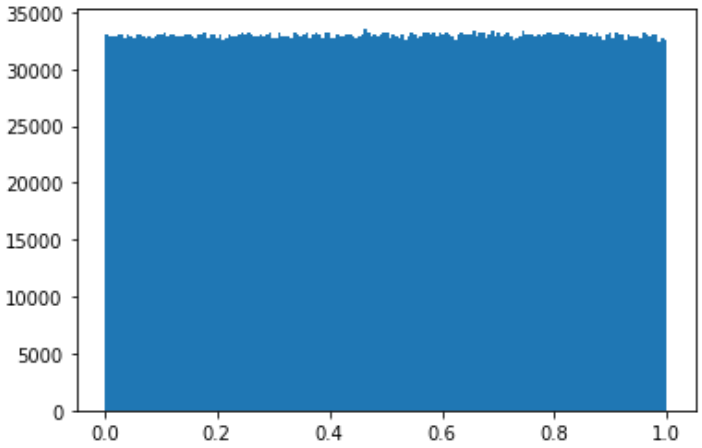



25000
20000
15000
10000
5000
0
0.0
0.2
0.4
0.6
0.8
1.0

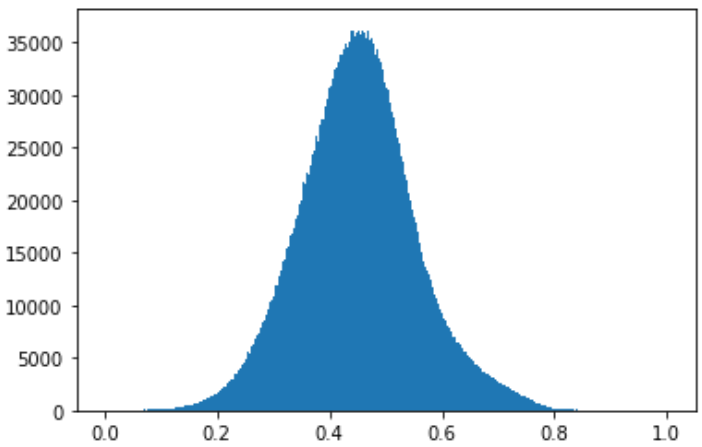
4


35000
30000
25000
20000
15000
10000
5000
0
0.0
0.2
0.4
0.6
0.8
1.0

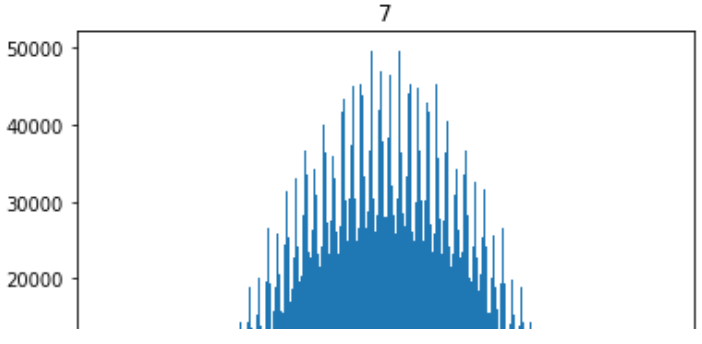
5

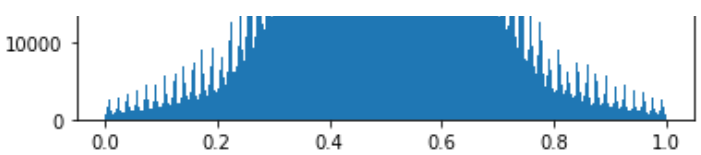

35000
30000
25000
20000
15000
10000
5000
0
0.0
0.2
0.4
0.6
0.8
1.0

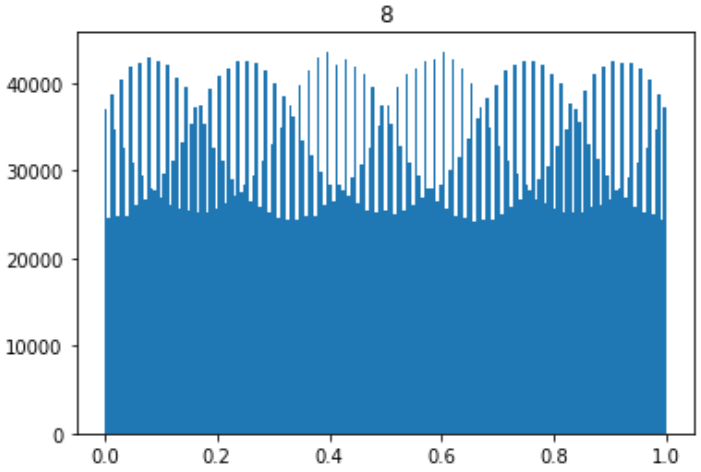
6

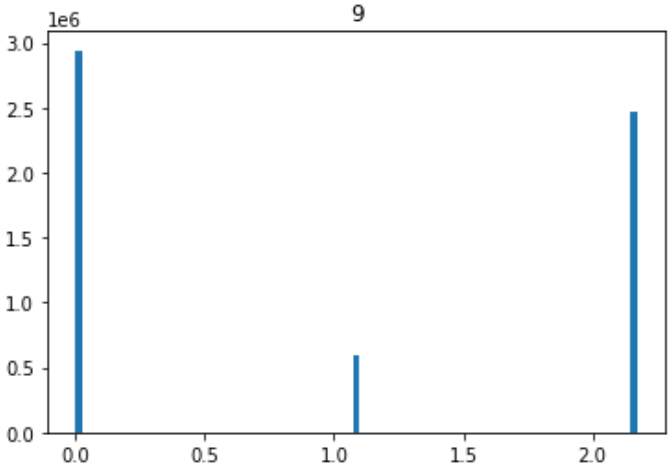

35000
30000
25000
20000
15000
10000
5000
0
0.0
0.2
0.4
0.6
0.8
1.0

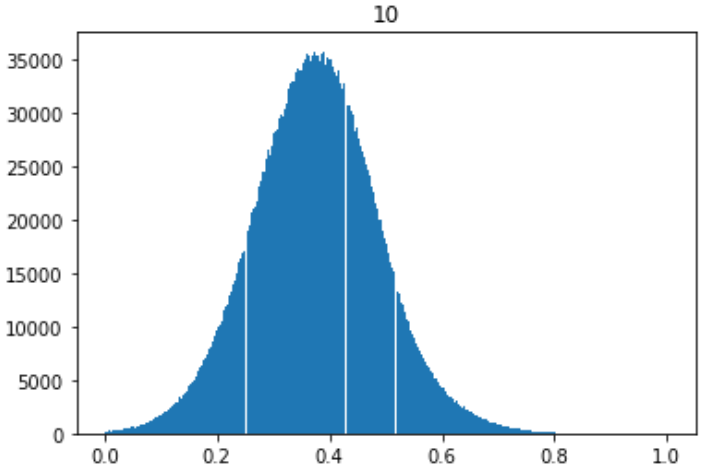
7

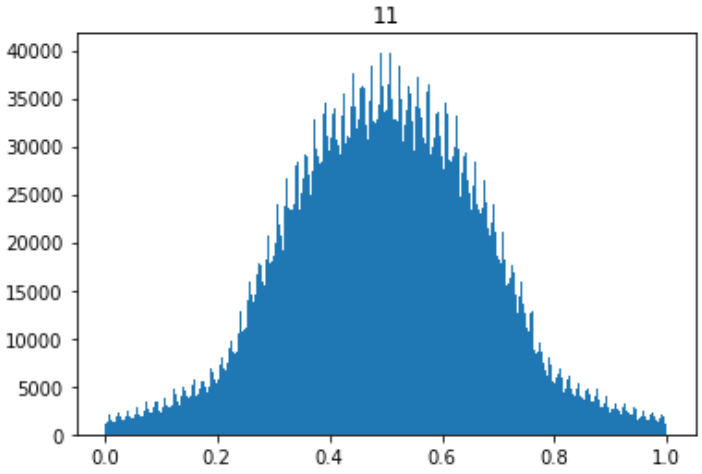

50000
40000
30000
20000

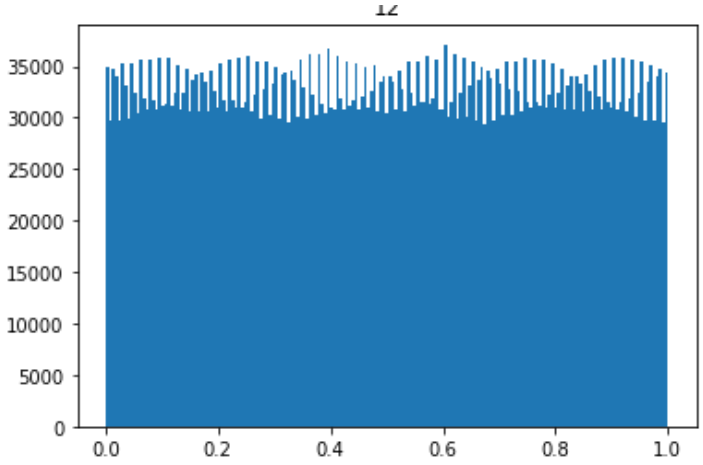

10000
0
0.0
0.2
0.4
0.6
0.8
1.0

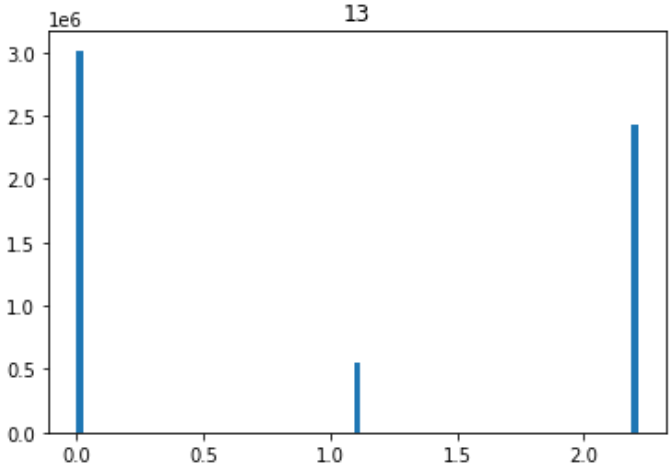

8
40000
30000
20000
10000
0
0.0
0.2
0.4
0.6
0.8
1.0

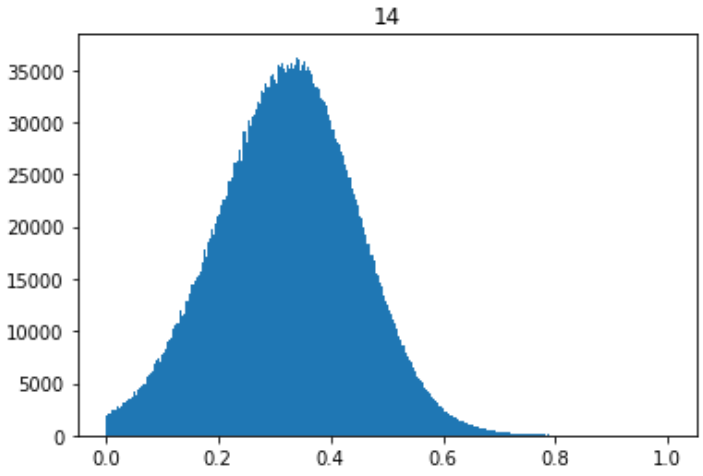

9
1e6
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0.0
0.5
1.0
1.5
2.0

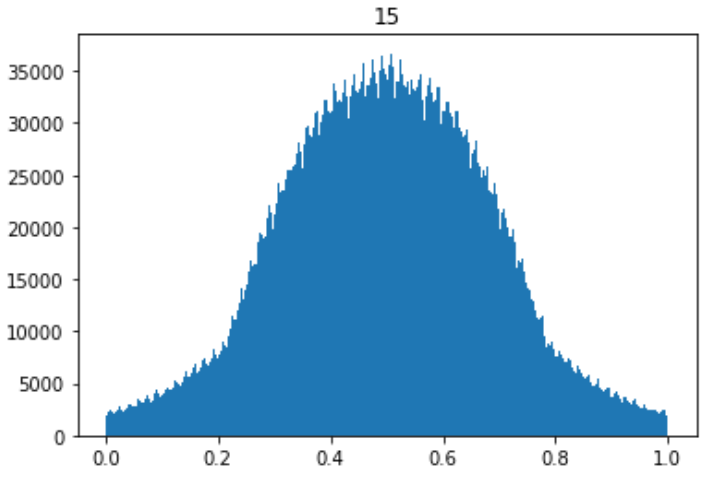

10
35000
30000
25000
20000
15000
10000
5000
0
0.0
0.2
0.4
0.6
0.8
1.0

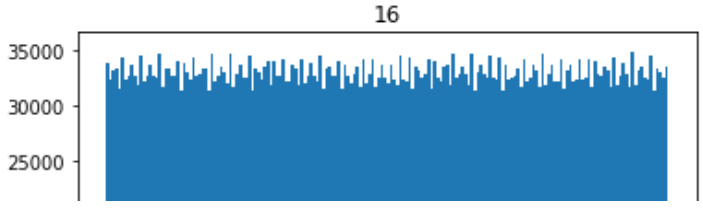

11
40000
35000
30000
25000
20000
15000
10000
5000
0
0.0
0.2
0.4
0.6
0.8
1.0

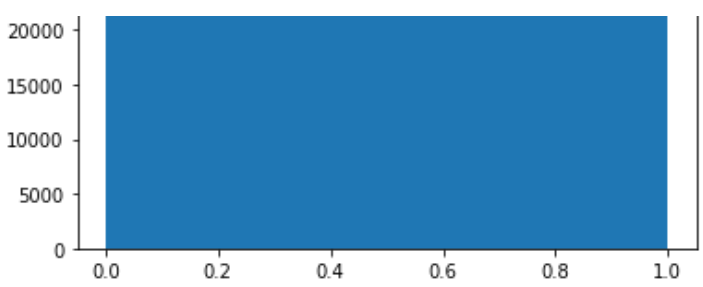

12
35000
30000
25000
20000
15000
10000
5000
0
0.0
0.2
0.4
0.6
0.8
1.0

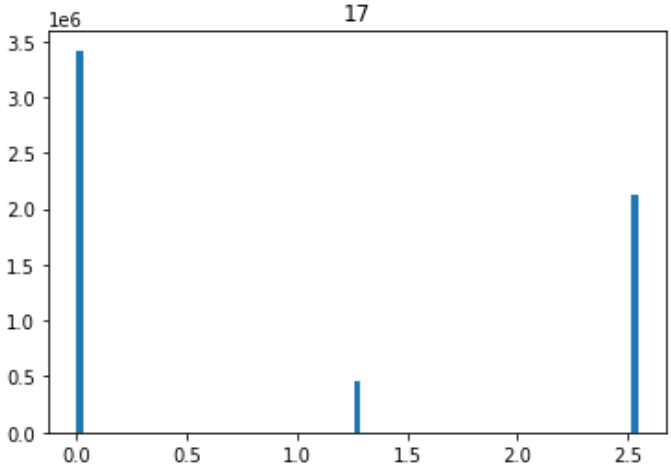

13
1e6
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0.0
0.5
1.0
1.5
2.0

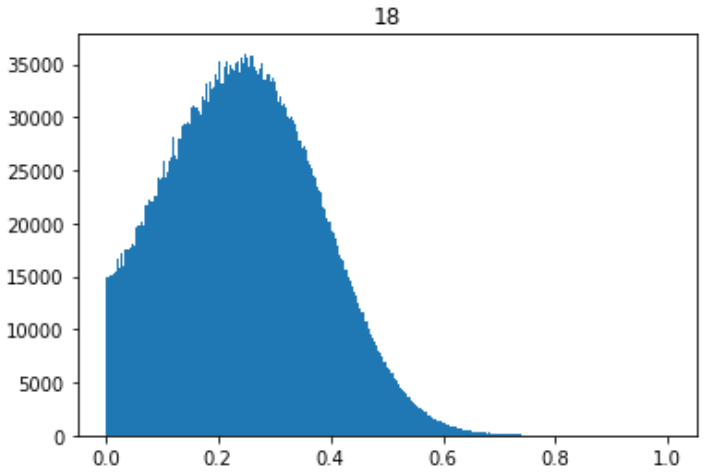

14
35000
30000
25000
20000
15000
10000
5000
0
0.0
0.2
0.4
0.6
0.8
1.0

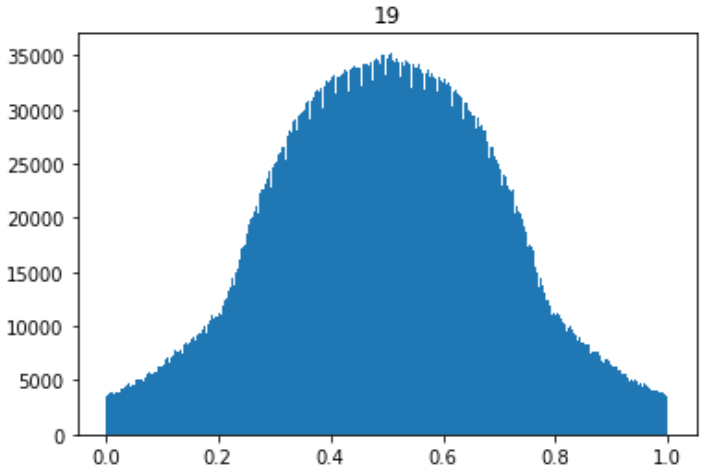

15
35000
30000
25000
20000
15000
10000
5000
0
0.0
0.2
0.4
0.6
0.8
1.0

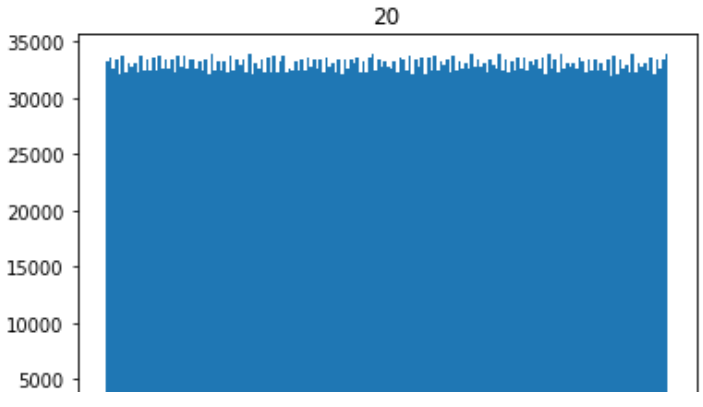

16
35000
30000
25000

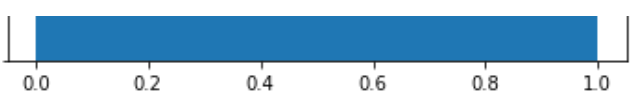

20000
15000
10000
5000
0
0.0
0.2
0.4
0.6
0.8
1.0

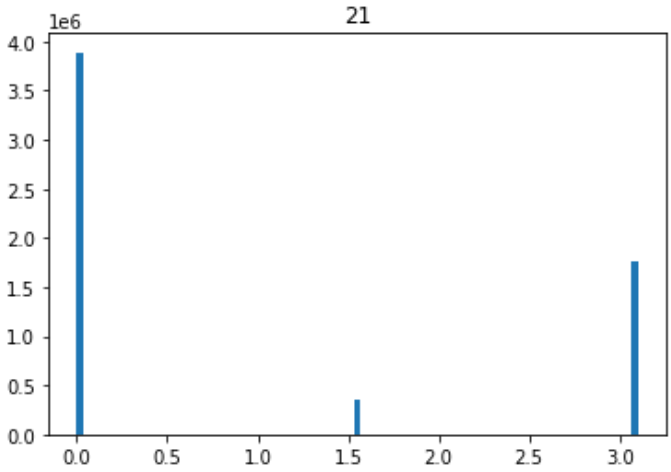

17
1e6
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0.0
0.5
1.0
1.5
2.0
2.5

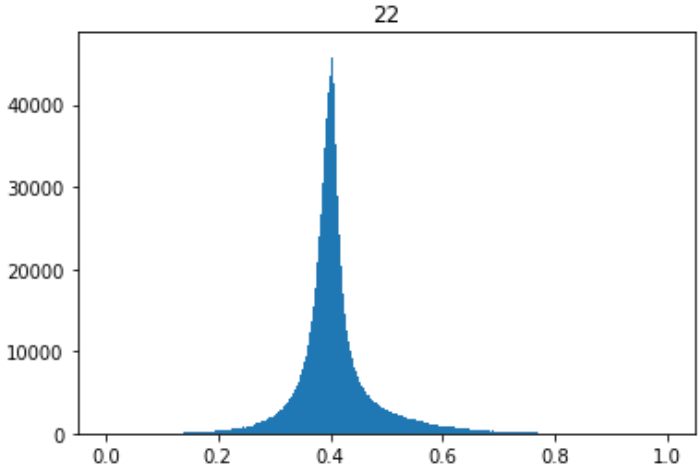

18
35000
30000
25000
20000
15000
10000
5000
0
0.0
0.2
0.4
0.6
0.8
1.0

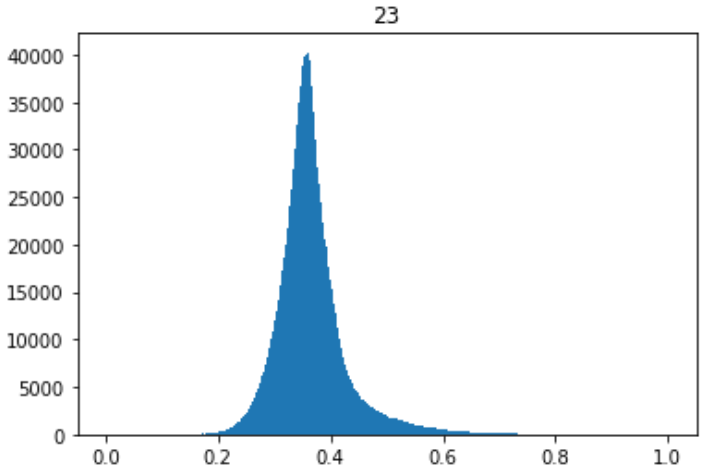

19
35000
30000
25000
20000
15000
10000
5000
0
0.0
0.2
0.4
0.6
0.8
1.0

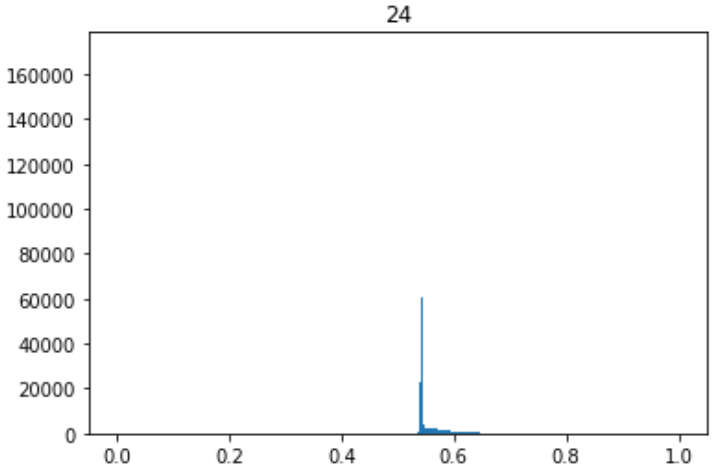

20
35000
30000
25000
20000
15000
10000
5000

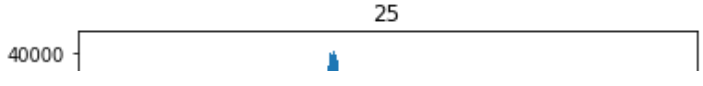

0
0.0
0.2
0.4
0.6
0.8
1.0

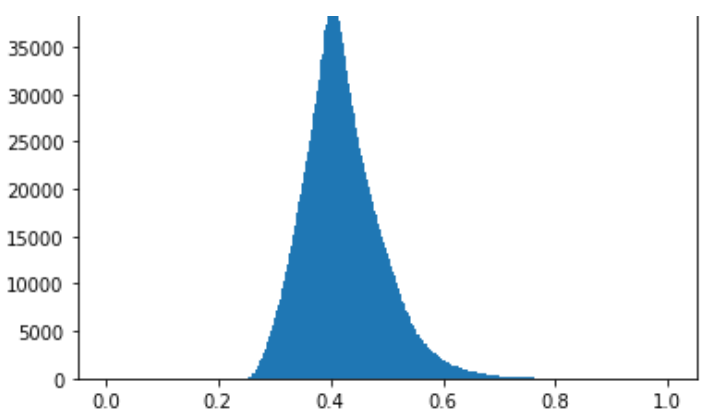

21
1e6
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0.0
0.5
1.0
1.5
2.0
2.5
3.0

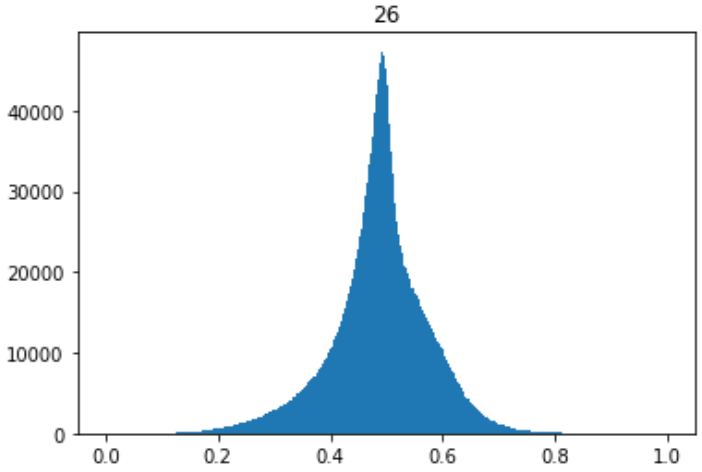

22
40000
30000
20000
10000
0
0.0
0.2
0.4
0.6
0.8
1.0

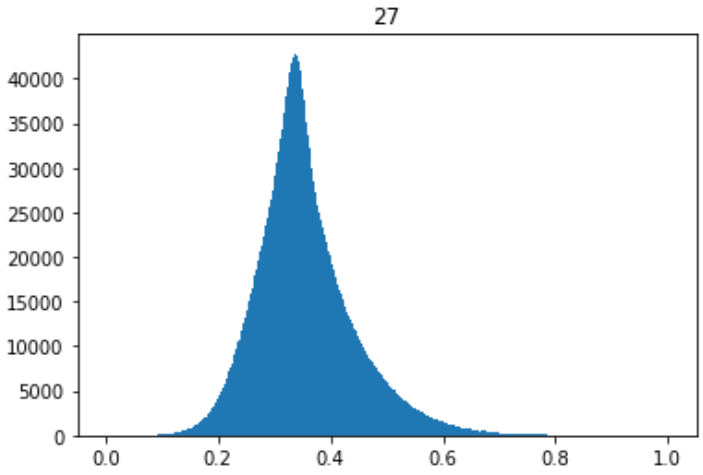

23
40000
35000
30000
25000
20000
15000
10000
5000
0
0.0
0.2
0.4
0.6
0.8
1.0

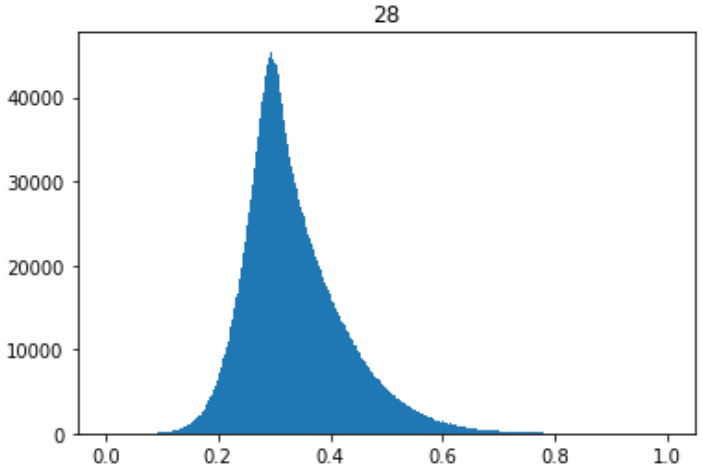

24
160000
140000
120000
100000
80000
60000
40000
20000
0
0.0
0.2
0.4
0.6
0.8
1.0


25
40000









In [ ]:

```
for index in categorical:
  mod_dataset[:,index]=mod_dataset[:,index].astype(int)
```

In [ ]:

```
X = mod_dataset[:,1:]
```

```
Y = mod_dataset[:,0].astype(int)
print(X[0],Y[0])
print(np.shape(X),np.shape(X[0]),np.shape(Y),np.shape(Y[0]))
```

```
[0.32890303 0.56759999 0.6030047  0.7796093  0.41033889 0.49819829 0.40613539 0.04393373
2.         0.35392154 0.46332112 0.86453768 2.         0.16984222 0.26892296 0.70997375
0.         0.0245159
 0.27192399 0.49966728 0.         0.21731784 0.32833051 0.54171222 0.42770875 0.47117561
0.35743408 0.28916102] 1
(5999999, 28) (28,) (5999999,) ()
```

In [ ]:

```
del mod_dataset
```

In [ ]:

```
train, test, y_train, y_test = train_test_split(X,Y, test_size=0.01)
train_x = train
test_x = test
train_y = train_x
test_y = test_x


train_ds = TensorDataset(torch.tensor(train_x, dtype=torch.float), torch.tensor(train_y,
dtype=torch.float))
test_ds = TensorDataset(torch.tensor(test_x, dtype=torch.float), torch.tensor(test_y, dt
ype=torch.float))

bs = 256

train_dl = DataLoader(train_ds, batch_size=bs, shuffle=True)
test_dl = DataLoader(test_ds, batch_size=bs * 2)
dls = core.DataLoaders(train_dl, test_dl)
```

Modelling a 5 layer DAE on this processed dataset

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.tan
h(self.en1(x)))))))))

    def decode(self, x):
        return self.de5(self.tanh(self.de4(self.tanh(self.de3(self.tanh(self.de2(self.tan
h(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()
)
```

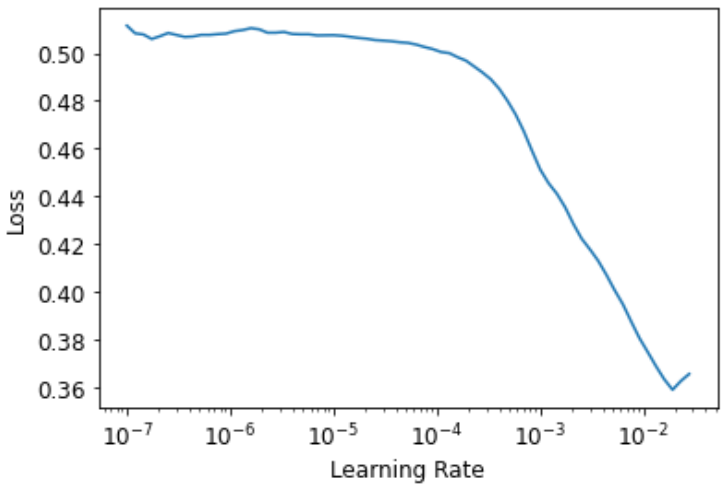
In [ ]:

```
loss_func = nn.MSELoss()

#bn_wd = False  # Don't use weight decay for batchnorm layers
true_wd = True  # weight decay will be used for all optimizers
wd = 1e-6

recorder = learner.Recorder()
learn = learner.Learner(dls, model=model, wd=wd, loss_func=loss_func, cbs=recorder)
```

In [ ]:

```
lr_min, lr_steep = learn.lr_find()

print('Learning rate with the minimum loss:', lr_min)
print('Learning rate with the steepest gradient:', lr_steep)
```

```
Learning rate with the minimum loss: 0.0019054606556892395
Learning rate with the steepest gradient: 0.0006918309954926372
```



In [ ]:

```
start = time.perf_counter()
learn.fit_one_cycle(n_epoch=6,lr_max=lr_min)
end = time.perf_counter()
delta_t = end - start
print('Training took', delta_t, 'seconds')
```

epoch	train_loss	valid_loss	time
0	0.006768	0.006965	04:59
0	0.006768	0.006965	04:59
1	0.005052	0.005163	04:57
1	0.005052	0.005163	04:57
2	0.004405	0.004531	05:06
2	0.004405	0.004531	05:06

epoch	train_loss	valid_loss	time
3	0.004100	0.004148	04:58
3	0.004100	0.004148	04:58
4	0.003793	0.003810	04:57
4	0.003793	0.003810	04:57
5	0.003634	0.003651	04:55
5	0.003634	0.003651	04:55

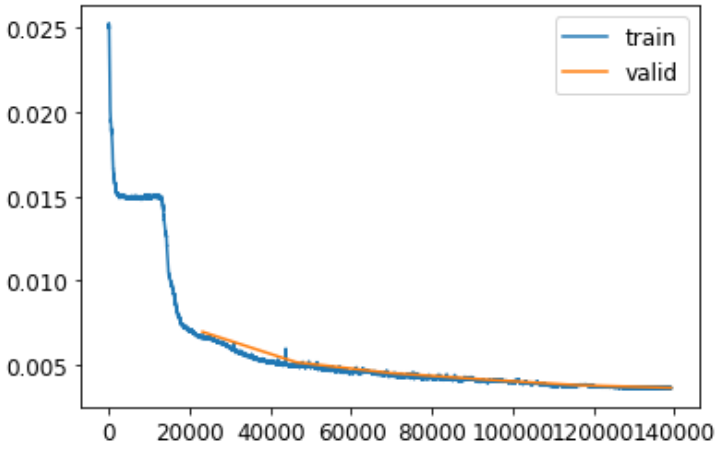
```
Training took 1794.9892121070002 seconds
```

In [ ]:

```
recorder.plot_loss()
learn.validate()
```

Out[ ]:

```
(#1) [0.0036510382778942585]
```



In [ ]:

```
save_dir = "dae_model"
if not os.path.exists(save_dir):
    os.makedirs(save_dir)
```

In [ ]:

```
torch.save(model.state_dict(), "dae_model/model.pth")
```

Load 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>
```

In [ ]:

```
data = torch.tensor(X[:10**4], dtype=torch.float)

pred = model_inf(data)
pred = pred.detach().numpy()
data = data.detach().numpy()

data_df = pd.DataFrame(data)
```

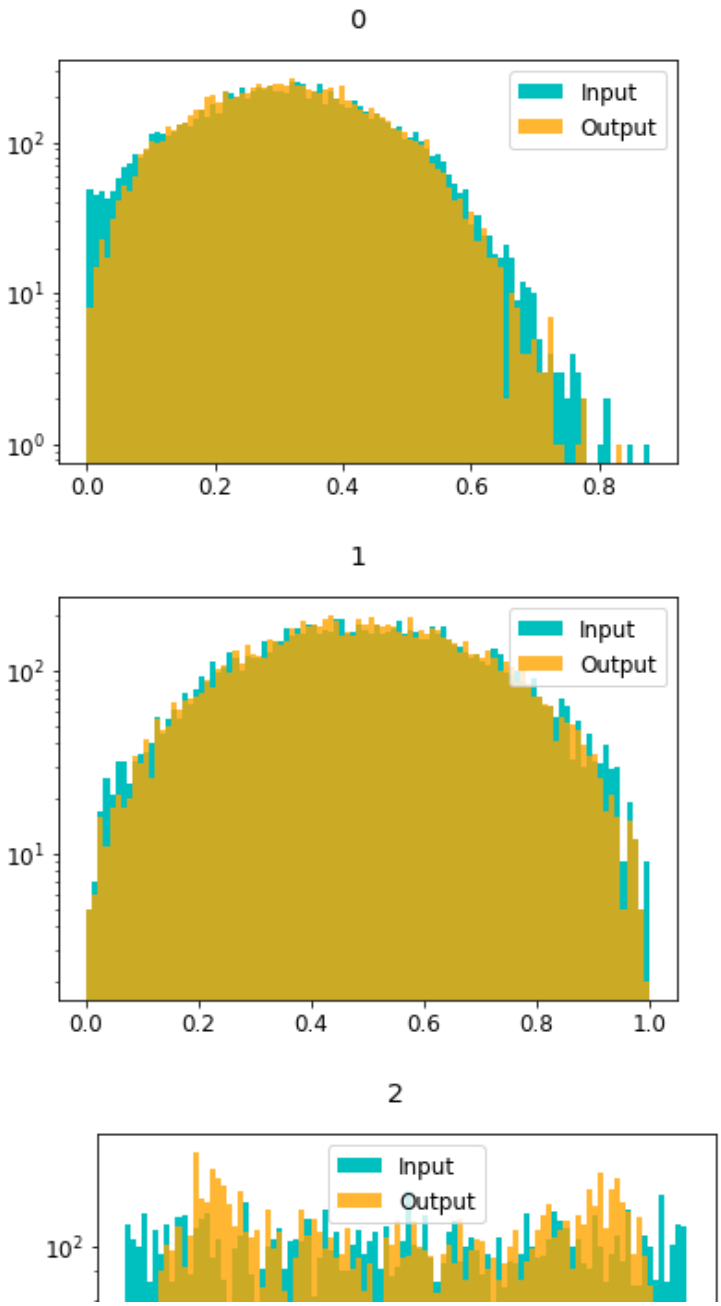
```
pred_df = pd.DataFrame(pred)
```

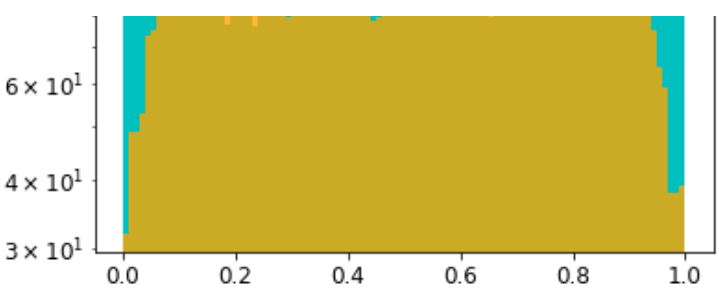
Results

In [ ]:

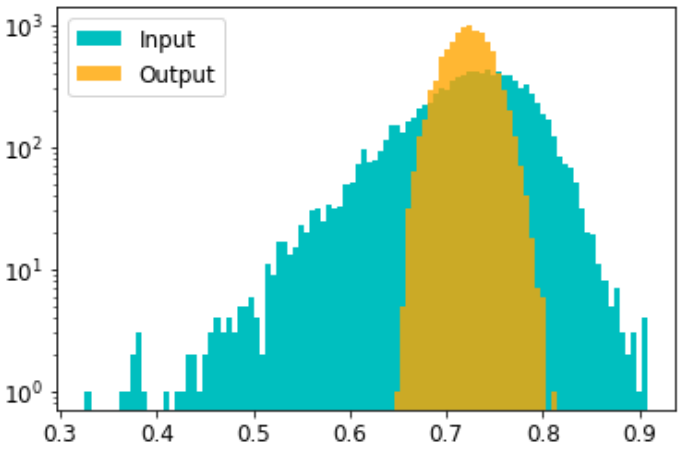
```
alph = 0.8
n_bins = 100
colors = ['orange', 'c']

for kk in np.arange(28):
    plt.figure()
    n_hist_data, bin_edges, _ = plt.hist(data[:, kk], color=colors[1], label='Input', al
pha=1, bins=n_bins)
    n_hist_pred, _, _ = plt.hist(pred[:, kk], color=colors[0], label='Output', alpha=alp
h, bins=bin_edges)
    plt.suptitle(str(kk))
    plt.yscale('log')
    # if True:
    #     plt.savefig(os.path.join(save_dir,str(kk)+'.png'))
    plt.legend()
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:6: RuntimeWarning: More than
20 figures have been opened. Figures created through the pyplot interface (`matplotlib.py
plot.figure`) are retained until explicitly closed and may consume too much memory. (To c
ontrol this warning, see the rcParam `figure.max_open_warning`).
```

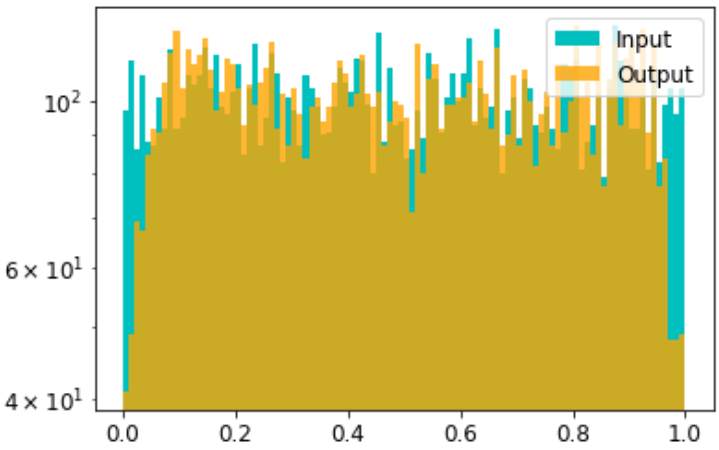



6×10^1
4×10^1
3×10^1
0.0
0.2
0.4
0.6
0.8
1.0

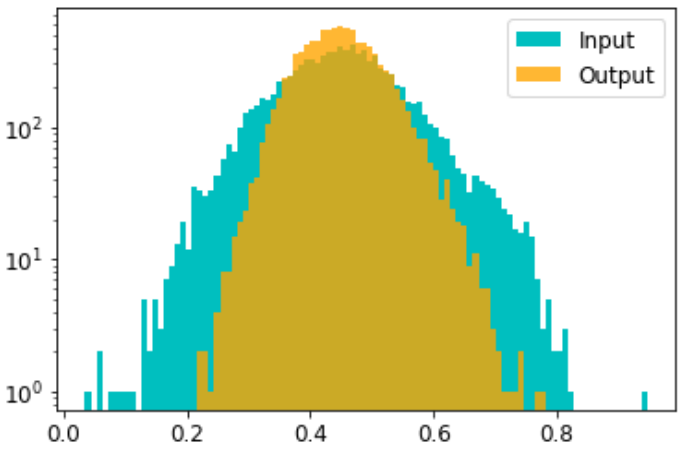
3


Input
Output
10^3
10^2
10^1
10^0
0.3
0.4
0.5
0.6
0.7
0.8
0.9

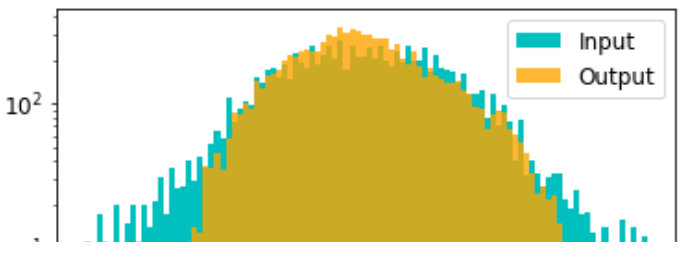
4

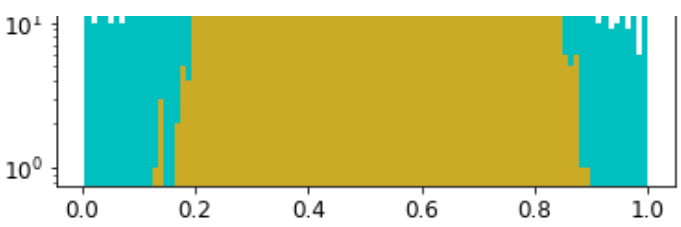

Input
Output
10^2
6×10^1
4×10^1
0.0
0.2
0.4
0.6
0.8
1.0

5

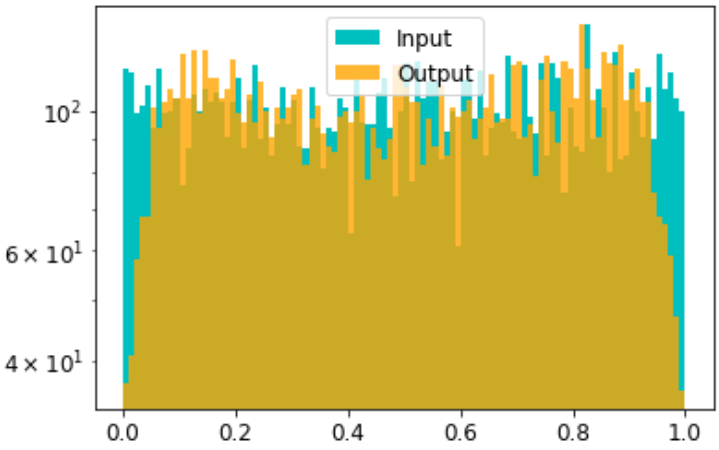

Input
Output
10^2
10^1
10^0
0.0
0.2
0.4
0.6
0.8

6

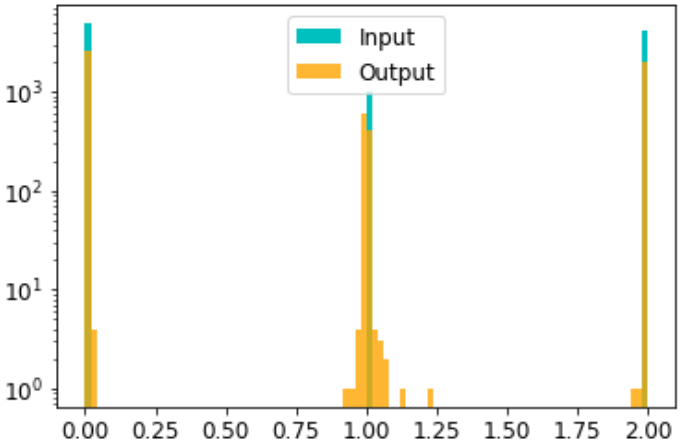

Input
Output
10^2


10^1
10^0
0.0
0.2
0.4
0.6
0.8
1.0

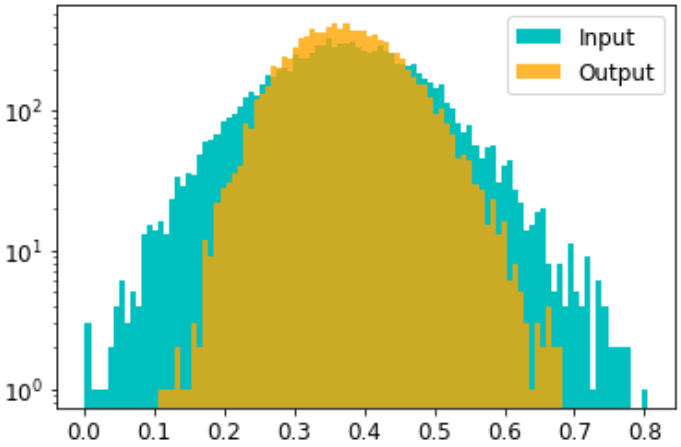
7


Input
Output
10^2
6×10^1
4×10^1
0.0
0.2
0.4
0.6
0.8
1.0

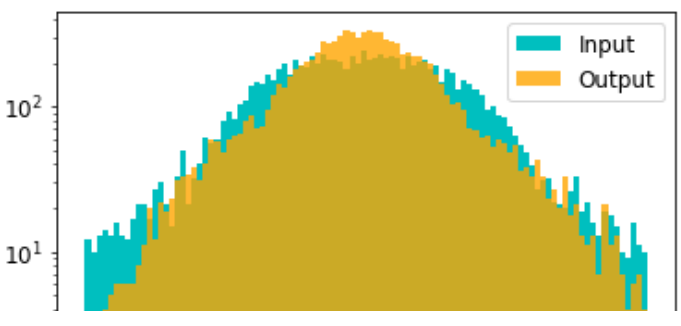
8

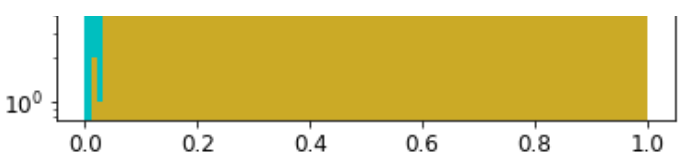

Input
Output
10^3
10^2
10^1
10^0
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00

9

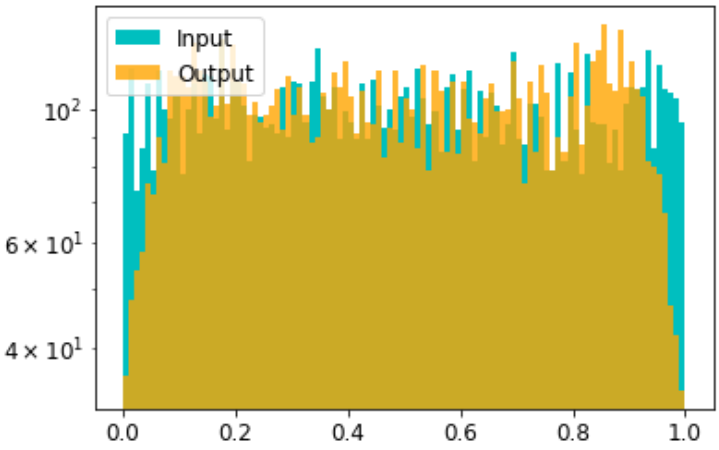

Input
Output
10^2
10^1
10^0
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8

10

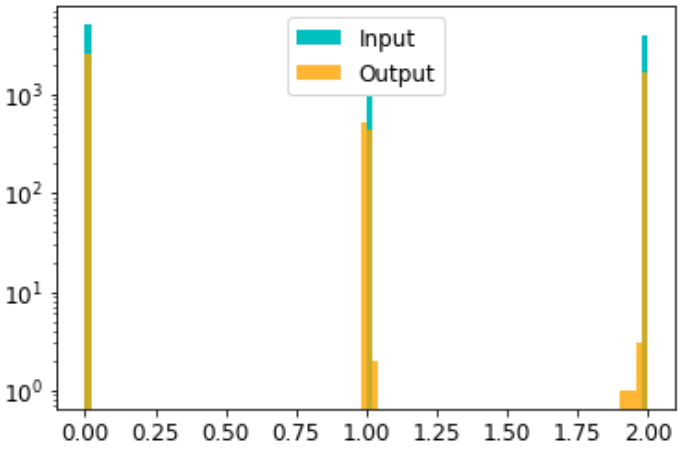

Input
Output
10^2
10^1


10^0
0.0
0.2
0.4
0.6
0.8
1.0

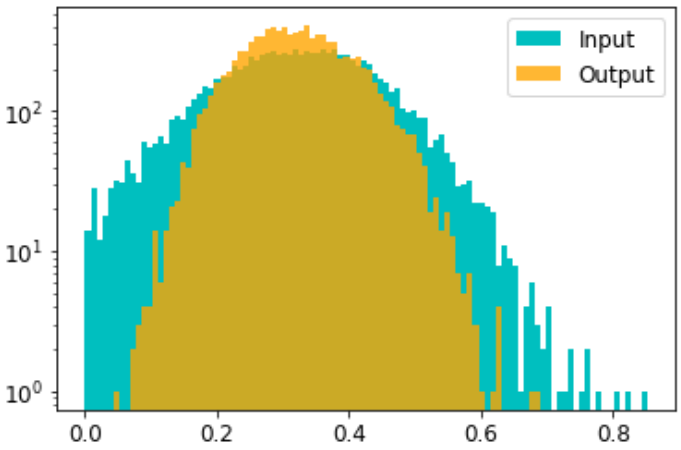
11


Input
Output
10^2
6×10^1
4×10^1
0.0
0.2
0.4
0.6
0.8
1.0

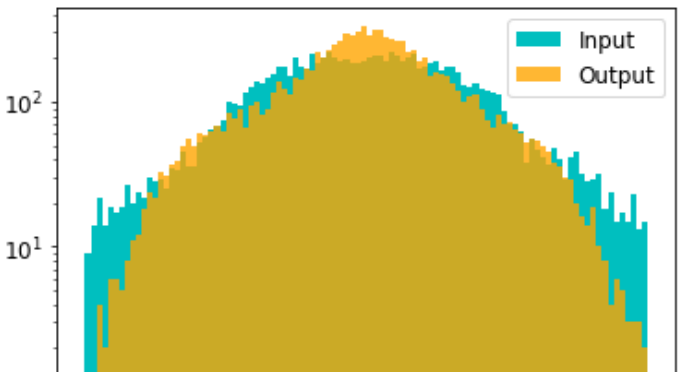
12

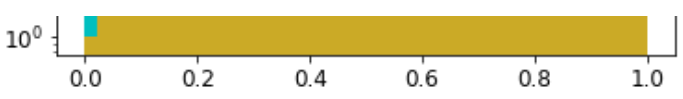

Input
Output
10^3
10^2
10^1
10^0
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00

13

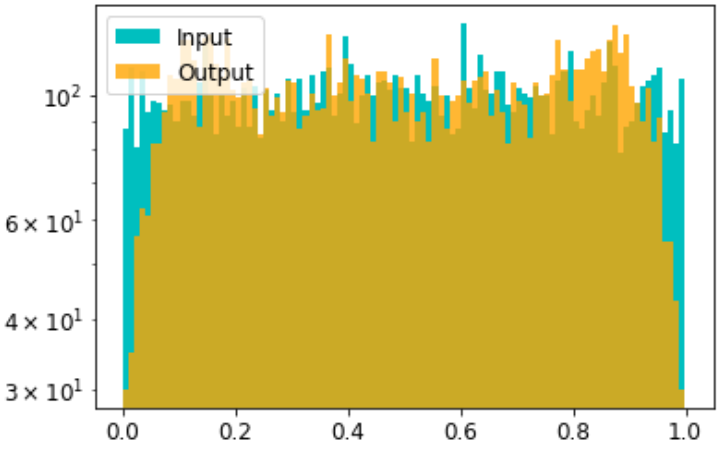

Input
Output
10^2
10^1
10^0
0.0
0.2
0.4
0.6
0.8

14

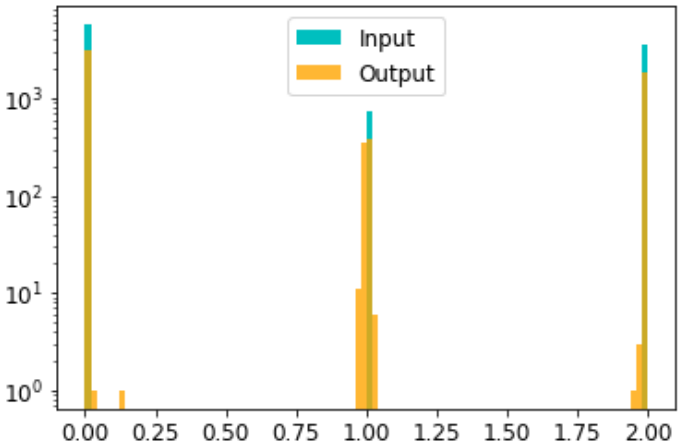

Input
Output
10^2
10^1


10^0
0.0
0.2
0.4
0.6
0.8
1.0

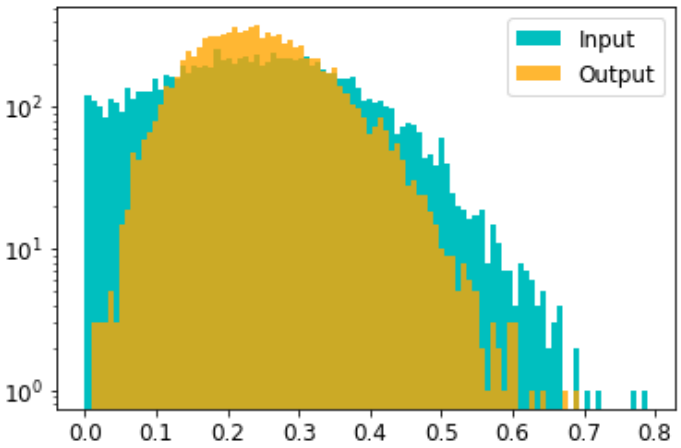
15


Input
Output
10^2
6×10^1
4×10^1
3×10^1
0.0
0.2
0.4
0.6
0.8
1.0

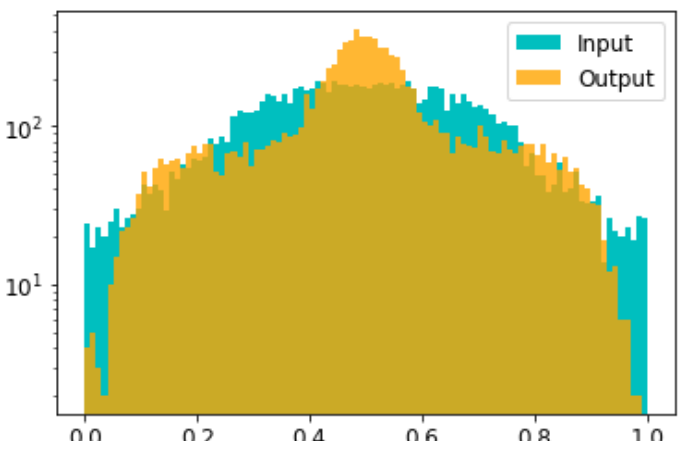
16


Input
Output
10^3
10^2
10^1
10^0
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00

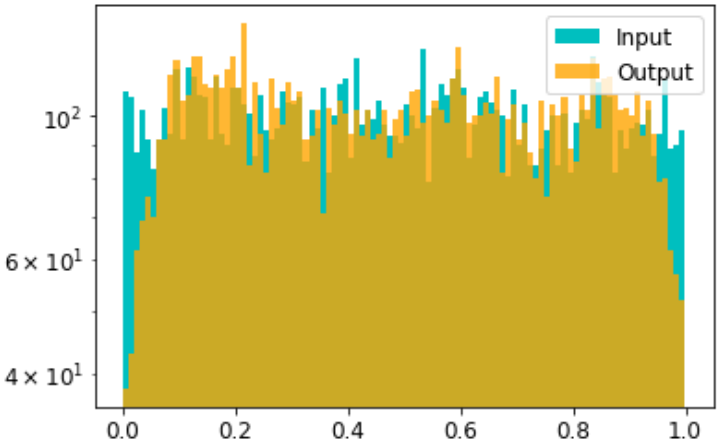
17


Input
Output
10^2
10^1
10^0
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8

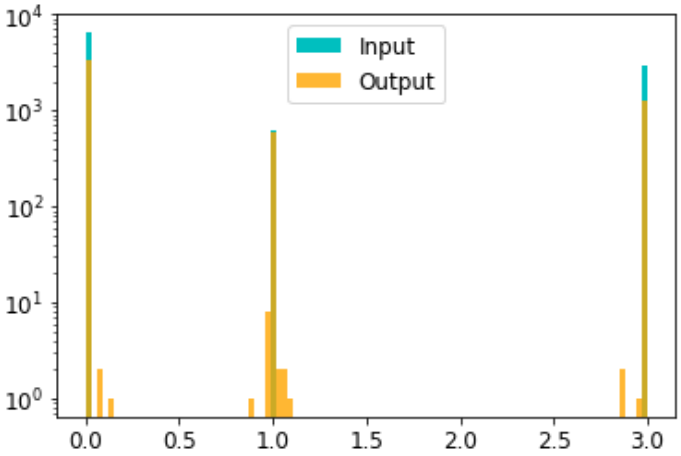
18


Input
Output
10^2
10^1
0.0
0.2
0.4
0.6
0.8
1.0

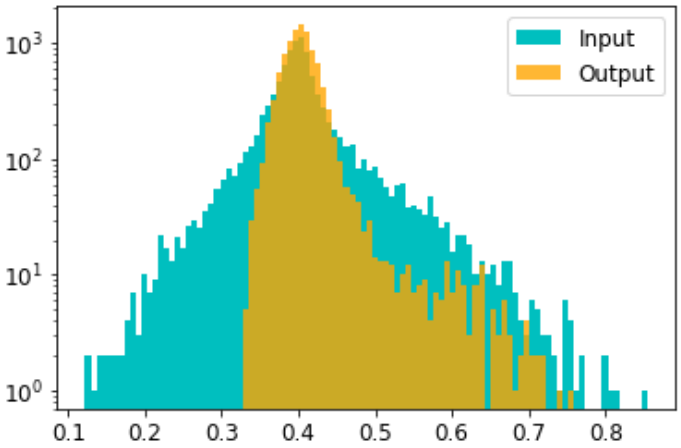
19


Input
Output
10^2
6×10^1
4×10^1
0.0
0.2
0.4
0.6
0.8
1.0

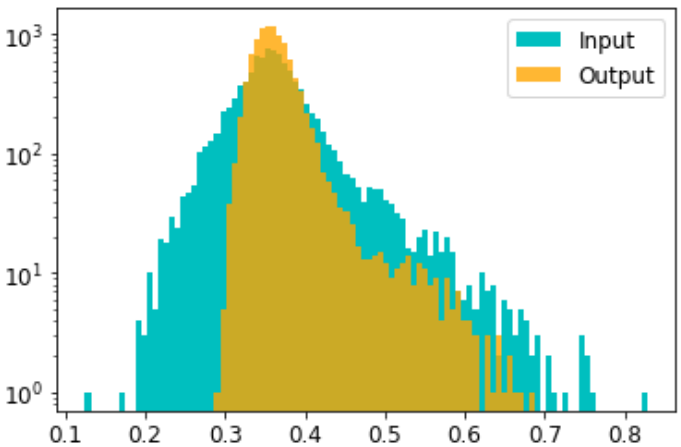
20


Input
Output
10^4
10^3
10^2
10^1
10^0
0.0
0.5
1.0
1.5
2.0
2.5
3.0

21

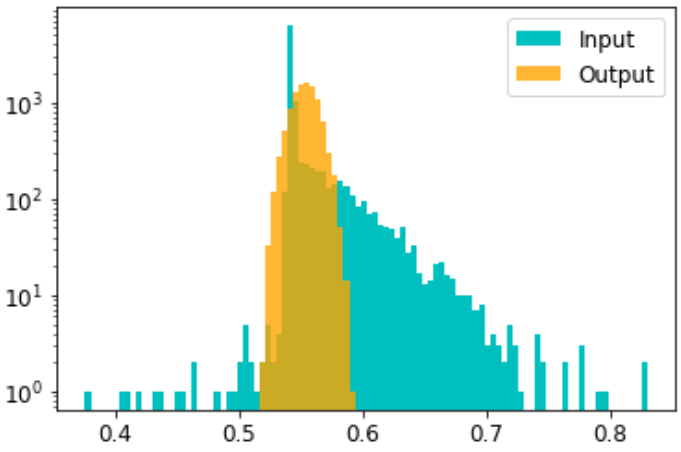

Input
Output
10^3
10^2
10^1
10^0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8

22

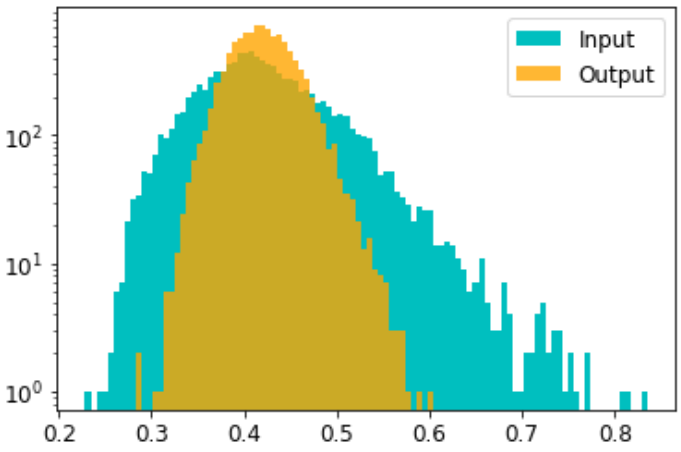

Input
Output
10^3
10^2
10^1
10^0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8

23

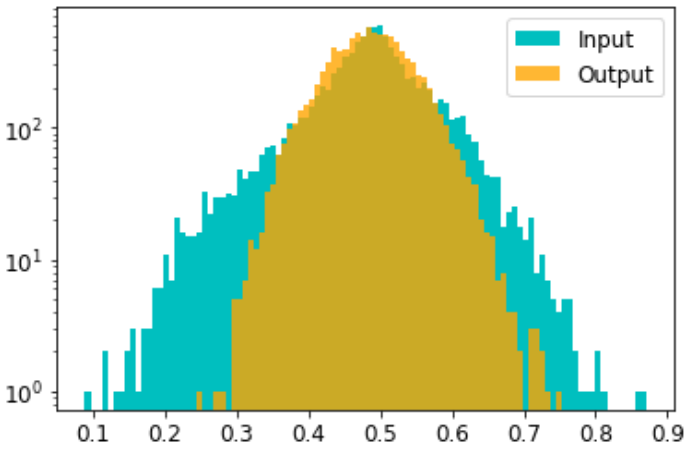
23


Input
Output
10^3
10^2
10^1
10^0
0.4
0.5
0.6
0.7
0.8

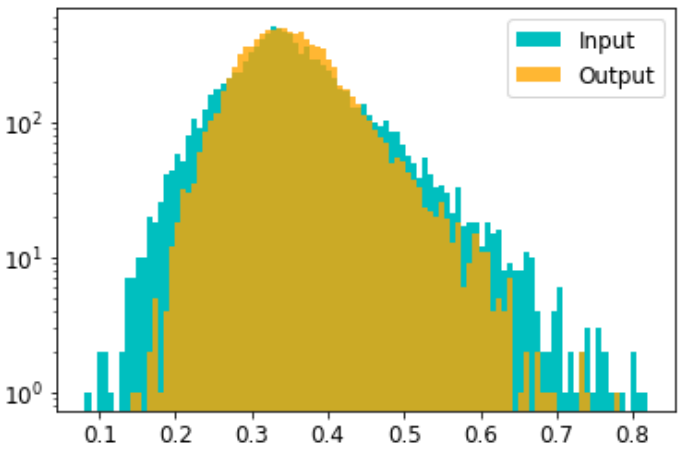
24


Input
Output
10^2
10^1
10^0
0.2
0.3
0.4
0.5
0.6
0.7
0.8

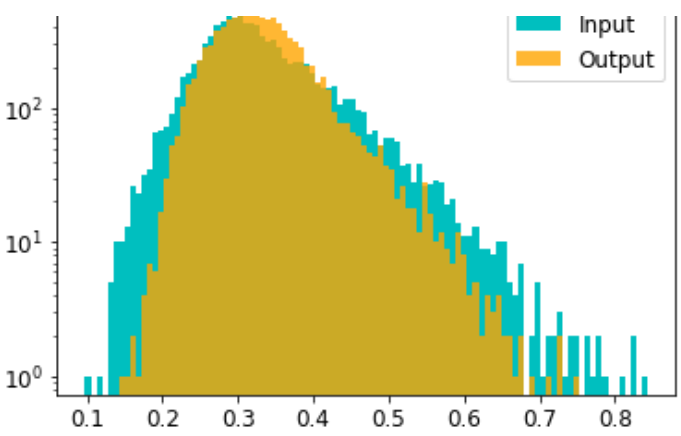
25


Input
Output
10^2
10^1
10^0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9

26


Input
Output
10^2
10^1
10^0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8

27


Input
Output
10^2
10^1
10^0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8