```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
import mpl_toolkits
```

```
from sklearn.datasets import make_blobs
```

```
data = make_blobs(n_samples=30000,
                  n_features= 9,
                  centers=8,
                  cluster_std=1.5,random_state=101)
```

```
data
```

```
(array([[-2.45357209, -8.38403325, -1.4550958 , ..., -2.88867807,
          7.69618161,  9.9188668 ],
        [ 7.14000781,  7.66401501, -2.46768811, ...,  4.87787766,
          8.50523118, -3.9998151 ],
        [-0.20715154,  2.52109171, -8.99126942, ..., -5.54586878,
          8.3421959 ,  4.32808467],
        ...,
        [ 2.8114073 ,  2.87094553, -8.26347635, ..., -2.88827882,
          9.91060705,  3.93479238],
        [-0.53498227, -6.35601343, -0.53456686, ..., -0.50481636,
         10.0566604 ,  8.48590727],
        [ 0.32301358, -7.14459862, -0.85037136, ...,  2.67963439,
          7.89082701,  8.98114608]]),
 array([6, 4, 0, ..., 0, 6, 6]))
```

```
X,y = data
```

```
X
```

```
array([[-2.45357209, -8.38403325, -1.4550958 , ..., -2.88867807,
         7.69618161,  9.9188668 ],
       [ 7.14000781,  7.66401501, -2.46768811, ...,  4.87787766,
         8.50523118, -3.9998151 ],
       [-0.20715154,  2.52109171, -8.99126942, ..., -5.54586878,
         8.3421959 ,  4.32808467],
       ...,
       [ 2.8114073 ,  2.87094553, -8.26347635, ..., -2.88827882,
```

```
         9.91060705,  3.93479238],
       [-0.53498227, -6.35601343, -0.53456686, ..., -0.50481636,
        10.0566604 ,  8.48590727],
       [ 0.32301358, -7.14459862, -0.85037136, ...,  2.67963439,
         7.89082701,  8.98114608]])
```

```
y
```

```
array([6, 4, 0, ..., 0, 6, 6])
```

```
np.random.seed(seed = 101)
z_noise = np.random.normal(size = len(X))
z_noise = pd.Series(z_noise)
```

```
z_noise
```

```
0        2.706850
1        0.628133
2        0.907969
3        0.503826
4        0.651118
           ...
29995   -0.576432
29996    1.707118
29997   -2.421848
29998    0.542748
29999    0.860208
Length: 30000, dtype: float64
```

```
feat = pd.DataFrame(X)
```

```
feat
```

	0	1	2	3	4	5	6	7	8
0	−2.453572	−8.384033	−1.455096	−9.058411	−10.476234	−5.373996	−2.888678	7.696182	9.918867
1	7.140008	7.664015	−2.467688	9.726943	−5.467735	−9.010938	4.877878	8.505231	−3.999815
2	−0.207152	2.521092	−8.991269	−6.080918	1.814173	4.615503	−5.545869	8.342196	4.328085
3	0.826518	4.990871	2.115152	8.023578	6.110141	−2.242148	1.244624	11.196717	−0.562374
4	5.703535	7.307323	0.649265	7.758733	−7.125417	−9.380699	2.165686	6.867014	−7.184363
…	…	…	…	…	…	…	…	…	…
29995	4.947985	9.105183	0.014771	7.747455	−5.282873	−7.377595	2.786648	8.123265	−7.509797
29996	1.108130	3.926287	2.424675	7.124342	6.569129	−2.036772	−3.489134	10.626713	−4.602558
29997	2.811407	2.870946	−8.263476	−8.501385	5.045543	8.086509	−2.888279	9.910607	3.934792
29998	−0.534982	−6.356013	−0.534567	−7.461386	−9.113174	−6.824077	−0.504816	10.056660	8.485907
29999	0.323014	−7.144599	−0.850371	−6.877320	−8.636506	−8.950869	2.679634	7.890827	8.981146

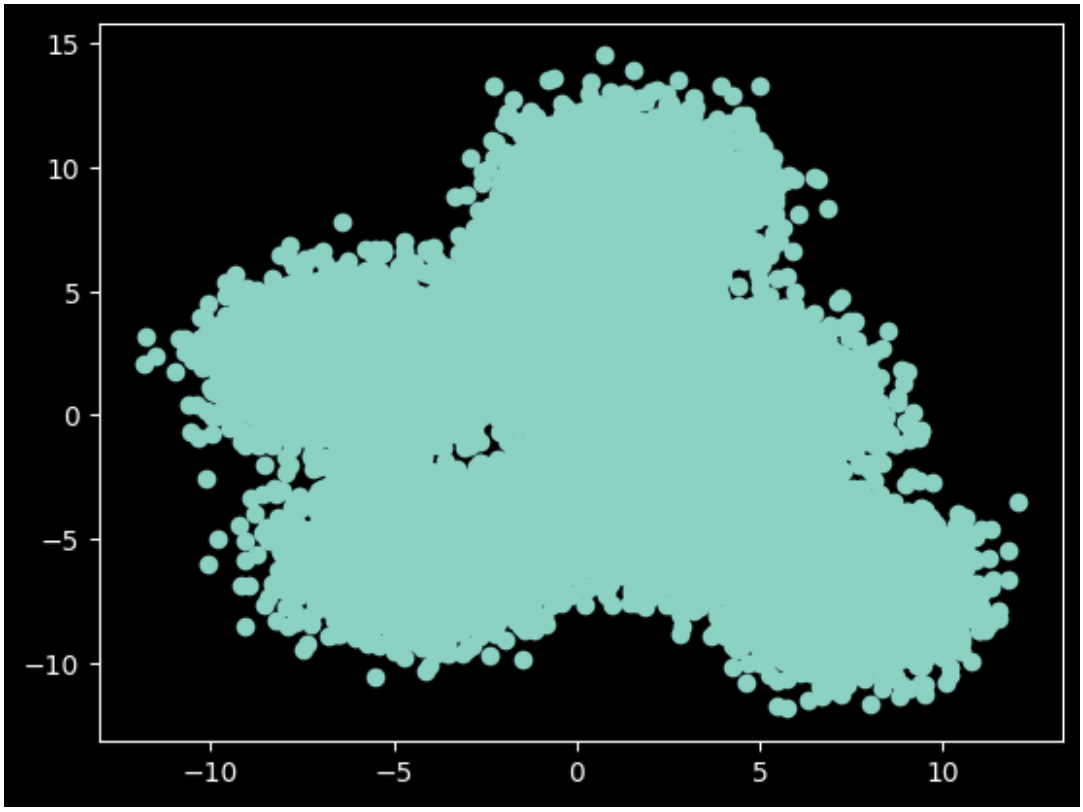
```
feat = pd.concat([feat,z_noise],axis=1)
```

```
feat.columns = ['X1','X2','X3','X4','X5','X6','X7','X8','X9','X10']
```

```
feat
```

	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10
0	−2.453572	−8.384033	−1.455096	−9.058411	−10.476234	−5.373996	−2.888678	7.696182	9.918867	2.706850
1	7.140008	7.664015	−2.467688	9.726943	−5.467735	−9.010938	4.877878	8.505231	−3.999815	0.628133
2	−0.207152	2.521092	−8.991269	−6.080918	1.814173	4.615503	−5.545869	8.342196	4.328085	0.907969
3	0.826518	4.990871	2.115152	8.023578	6.110141	−2.242148	1.244624	11.196717	−0.562374	0.503826
4	5.703535	7.307323	0.649265	7.758733	−7.125417	−9.380699	2.165686	6.867014	−7.184363	0.651118
…	…	…	…	…	…	…	…	…	…	…
29995	4.947985	9.105183	0.014771	7.747455	−5.282873	−7.377595	2.786648	8.123265	−7.509797	0.576432
29996	1.108130	3.926287	2.424675	7.124342	6.569129	−2.036772	−3.489134	0.626713	−4.602558	1.707118
29997	2.811407	2.870946	−8.263476	−8.501385	5.045543	8.086509	−2.888279	9.910607	3.934792	−2.421848
29998	−0.534982	−6.356013	−0.534567	−7.461386	−9.113174	−6.824077	−0.504816	0.056660	8.485907	0.542748
29999	0.323014	−7.144599	−0.850371	−6.877320	−8.636506	−8.950869	2.679634	7.890827	8.981146	0.860208

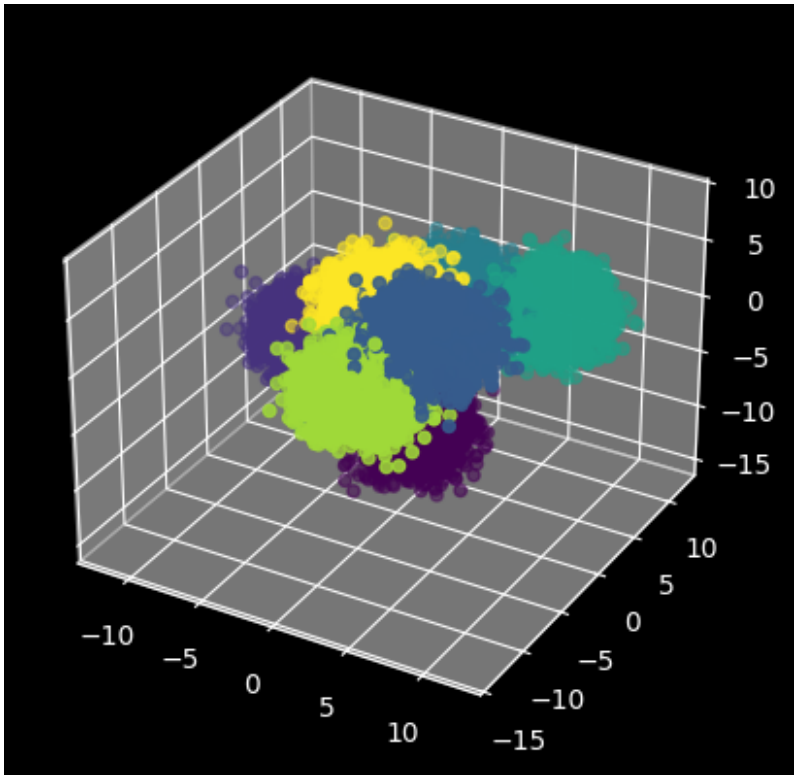
```
plt.scatter(feat['X1'],feat['X9'])
```




```
import mpl_toolkits
```

```
from mpl_toolkits.mplot3d import Axes3D
```

```
fig = plt.figure()
ax = fig.add_subplot(111, projection = '3d')
ax.scatter(feat['X1'],feat['X2'],feat['X3'], c = y)
```




```
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, Dense
from tensorflow.keras.optimizers import SGD
```

```
from tensorflow.keras.optimizers import SGD
```

```
# encoder = Sequential()
# encoder.add(Dense(units=8, activation='relu'))
# encoder.add(Dense(units=7, activation='relu'))
# encoder.add(Dense(units=3, activation='relu'))
```

```
# decoder = Sequential()
# decoder.add(Dense(units=5, activation='relu'))
# decoder.add(Dense(units=10, activation='relu'))
```

```
# autoencoder = Sequential([encoder, decoder])
```

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

```
scaler = MinMaxScaler()
scaled_data = scaler.fit_transform(feat)
```

```
# 10 --> 8 --> 7 --> 5 --> 3 --> 5 --> 8 --> 10
```

```
# Autoencoder model
input_layer = Input(shape=(scaled_data.shape[1],))
encoded = Dense(8, activation='relu')(input_layer)
encoded = Dense(7, activation='relu')(encoded)
encoded = Dense(5, activation='relu')(encoded)
encoded = Dense(3, activation='relu')(encoded)
decoded = Dense(5, activation='relu')(encoded)
decoded = Dense(8, activation='relu')(decoded)
decoded = Dense(10, activation='relu')(decoded)
decoded = Dense(scaled_data.shape[1], activation='relu')(decoded)

autoencoder = Model(inputs=input_layer, outputs=decoded)
autoencoder.compile(loss='mse', optimizer=SGD(learning_rate=1.0))
```

```
autoencoder.compile(loss = 'mse',
                    optimizer = SGD(learning_rate= 1.0))
```

```
# from sklearn.preprocessing import MinMaxScaler
```

```
# scaler = MinMaxScaler()
# scaled_data = scaler.fit_transform(feat)
```

```
# scaled_data
```

```
autoencoder.fit(scaled_data,
                scaled_data,
                batch_size = 256,
                shuffle = True,
                epochs = 12)
```

```
Epoch 1/12
118/118 ━━━━━━━━━━━━━━━━━━━━ 0s 571us/step - loss: 0.1806
Epoch 2/12
118/118 ━━━━━━━━━━━━━━━━━━━━ 0s 533us/step - loss: 0.1307
Epoch 3/12
118/118 ━━━━━━━━━━━━━━━━━━━━ 0s 677us/step - loss: 0.1221
```

```
Epoch 4/12
118/118 ━━━━━━━━━━━━━━━━━━━━ 0s 543us/step - loss: 0.1150
Epoch 5/12
118/118 ━━━━━━━━━━━━━━━━━━━━ 0s 541us/step - loss: 0.0712
Epoch 6/12
118/118 ━━━━━━━━━━━━━━━━━━━━ 0s 461us/step - loss: 0.0659
Epoch 7/12
118/118 ━━━━━━━━━━━━━━━━━━━━ 0s 487us/step - loss: 0.0640
Epoch 8/12
118/118 ━━━━━━━━━━━━━━━━━━━━ 0s 470us/step - loss: 0.0629
Epoch 9/12
118/118 ━━━━━━━━━━━━━━━━━━━━ 0s 454us/step - loss: 0.0625
Epoch 10/12
118/118 ━━━━━━━━━━━━━━━━━━━━ 0s 658us/step - loss: 0.0588
Epoch 11/12
118/118 ━━━━━━━━━━━━━━━━━━━━ 0s 712us/step - loss: 0.0373
Epoch 12/12
118/118 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.0371
```

```
<keras.src.callbacks.history.History at 0x3059c4e60>
```

```
encoder = Model(inputs=input_layer, outputs=encoded)
encoded_2dim = encoder.predict(scaled_data)
```

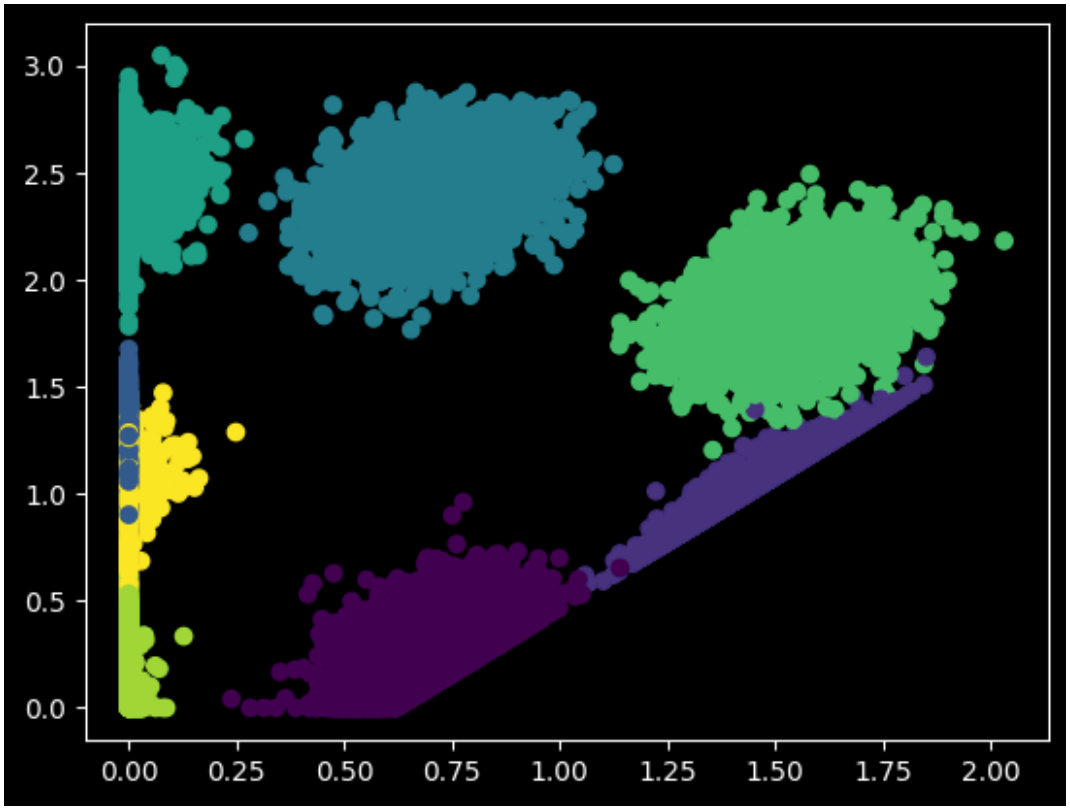
```
938/938 ━━━━━━━━━━━━━━━━━━━━ 0s 327us/step
```

```
decoder = Model(inputs = encoded, )
```

```
encoded_2dim
```

```
array([[0.        , 0.14576907, 0.        ],
       [0.        , 2.3666031 , 0.        ],
       [0.54025525, 0.30830514, 0.        ],
       ...,
       [0.7485703 , 0.3178265 , 0.        ],
       [0.        , 0.29478663, 0.        ],
       [0.        , 0.34100515, 0.        ]], dtype=float32)
```

```
plt.scatter(encoded_2dim[:,0],encoded_2dim[:,1], c = y)
```



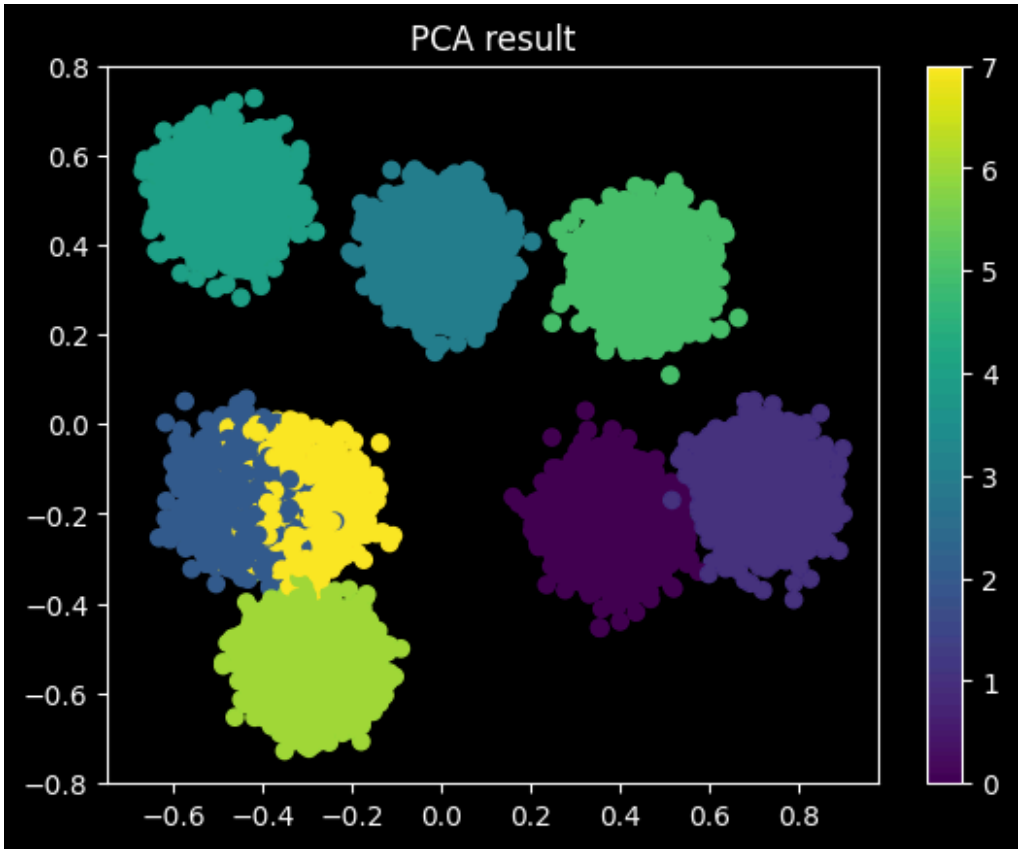

```
from sklearn.decomposition import PCA
```

```
pca = PCA(n_components=2)
pca_result = pca.fit_transform(scaled_data)
```

```
pca_result
```

```
array([[-0.20421224, -0.65802522],
       [-0.42493218,  0.52563915],
       [ 0.31452227, -0.21878356],
       ...,
       [ 0.3531953 , -0.21179669],
       [-0.30296668, -0.52045716],
       [-0.33961384, -0.48279011]])
```

```
plt.scatter(pca_result[:, 0], pca_result[:, 1], c=y, cmap='viridis')
plt.title('PCA result')
plt.colorbar()
plt.show()
```




```
from sklearn.metrics import mean_squared_error

X_reconstructed = pca.inverse_transform(pca_result)
```

```
mean_squared_error(X_reconstructed, scaled_data)
```

```
0.012899472591549787
```

```
X_reconstructed_auto_encoder =
```

```
mean_squared_error(encoded_2dim, scaled_data)
```

```
ValueError: y_true and y_pred have different number of output (3!=10)
```