Dataset2-Diabetes_output_2

November 3, 2021

1 Dataset 2 - Regression : Diabetes Dataset

1.1 Experiment Details

The aim of the experiment is to verify if the: 1. ABC_GAN model corrects model misspecification 2. ABC_GAN model performs better and converges faster than a simple C-GAN model

In the experiment we predict the distribution that represents the real data and simulate realistic fake data points using statistical mode, C-GAN and ABC-GAN model with 3 priors. We analyze and compare their performance using metrics like mean squared error, mean absolute error , manhattan distance and euclidean distance between y_{real} and y_{pred}

The models are as follows:

1. The statistical model assumes the distribution $Y = \beta X + \mu$ where $\mu \sim N(0,1)$
2. The Conditional GAN consists of
    1. Generator with 2 hidden layers with 100 nodes each and ReLu activation.
    2. Discriminator with 2 hidden layers with 25 and 50 nodes and ReLu activation. We use Adam's optimser and BCE Logit Loss to train the model. The input to the Generator of the GAN is (x,e) where x are the features and $e \sim N(0,1)$. The discriminator output is linear.
3. The ABC GAN Model consists of
    1. ABC generator is defined as follows:
        1. $Y = 1 + \beta_1 x_1 + \beta_2 x_2 + \beta_2 x_3 + ... + \beta_n x_n + N(0,\sigma)$ where $\sigma = 0.1$
        2. $\beta_i \sim N(0,\sigma^*)$ when $\mu = 0$ else $\beta_i \sim N(\beta_i^*, \sigma^*)$ where $\beta_i^* s$ are coefficients obtained from statistical model
        3. σ^* takes the values 0.01,0.1 and 1

    2. C-GAN network is as defined above. However the input to the Generator of the GAN is (x, y_{abc}) where y_{abc} is the output of the ABC Generator.

1.2 Import Libraries

[1]:
```
import warnings
import sys
sys.path.insert(0, '../src')
warnings.filterwarnings('ignore')
```

[2]:
```python
import train_test
import ABC_train_test
import diabetesDataset
import network
import statsModel
import performanceMetrics
import dataset
import sanityChecks
import torch
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import norm
from torch.utils.data import Dataset,DataLoader
from torch import nn
```

1.3 Parameters

ABC-Generator parameters are as mentioned below: 1. mean : 1 ($\beta \sim N(\beta^*, \sigma)$ where β^* are coefficients of statistical model) or 1 ($\beta \sim N(0, \sigma)$ 2. std : $\sigma = 1, 0.1, 0.01$ (standard deviation)

[3]:
```python
#ABC Generator Parameters
mean = 1
variance = 0.001
```

[4]:
```python
# Parameters
mean = 1
variance = 0.01
```

1.4 Dataset

The dataset used is sklearn's toy regression dataset : diabetes Properties: 1. 10 features 2. 442 datapoints

[5]:
```python
X,Y = diabetesDataset.diabetes_data()
n_samples = 442
n_features = 10
```

```
         X1        X2        X3        X4        X5        X6        X7  \
0  0.038076  0.050680  0.061696  0.021872 -0.044223 -0.034821 -0.043401
1 -0.001882 -0.044642 -0.051474 -0.026328 -0.008449 -0.019163  0.074412
2  0.085299  0.050680  0.044451 -0.005671 -0.045599 -0.034194 -0.032356
3 -0.089063 -0.044642 -0.011595 -0.036656  0.012191  0.024991 -0.036038
4  0.005383 -0.044642 -0.036385  0.021872  0.003935  0.015596  0.008142

         X8        X9       X10      Y
0 -0.002592  0.019908 -0.017646  151.0
1 -0.039493 -0.068330 -0.092204   75.0
2 -0.002592  0.002864 -0.025930  141.0
```

```
3  0.034309  0.022692 -0.009362  206.0
4 -0.002592 -0.031991 -0.046641  135.0
```

1.5 Stats Model

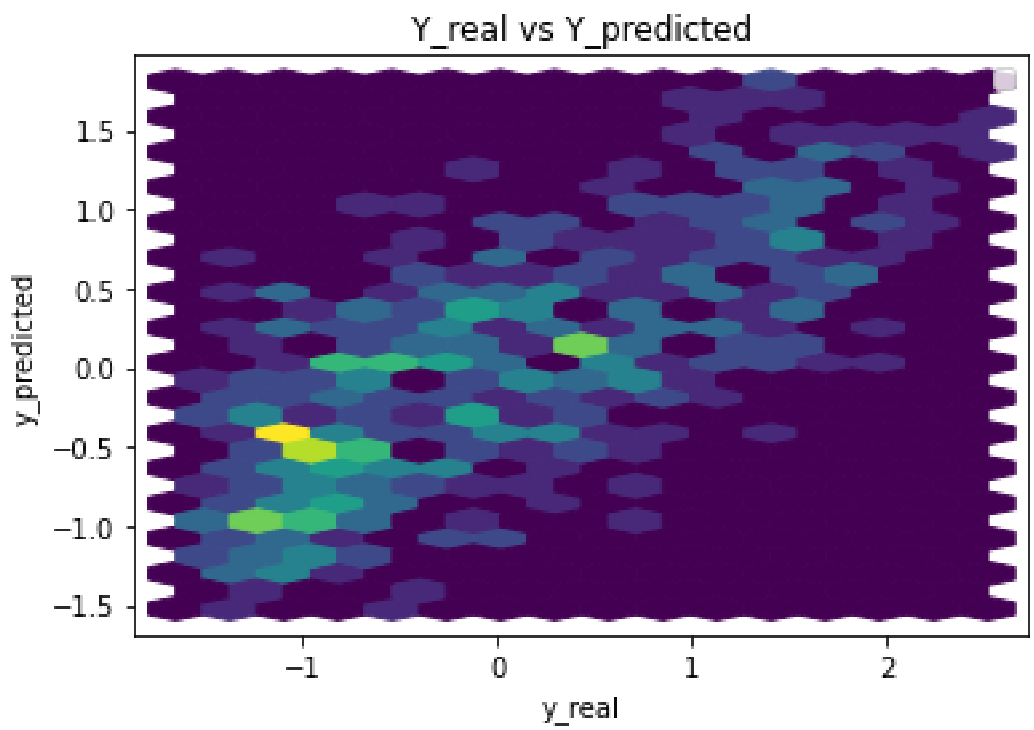
```
[6]: [coeff,y_pred] = statsModel.statsModel(X,Y)
```

```
No handles with labels found to put in legend.

                            OLS Regression Results
==============================================================================
Dep. Variable:                      Y   R-squared:                       0.518
Model:                            OLS   Adj. R-squared:                  0.507
Method:                 Least Squares   F-statistic:                     46.27
Date:                Wed, 03 Nov 2021   Prob (F-statistic):           3.83e-62
Time:                        19:51:52   Log-Likelihood:                -466.00
No. Observations:                 442   AIC:                             954.0
Df Residuals:                     431   BIC:                             999.0
Df Model:                          10
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const      -1.804e-16      0.033  -5.39e-15      1.000      -0.066       0.066
x1            -0.0062      0.037     -0.168      0.867      -0.079       0.066
x2            -0.1481      0.038     -3.917      0.000      -0.222      -0.074
x3             0.3211      0.041      7.813      0.000       0.240       0.402
x4             0.2004      0.040      4.958      0.000       0.121       0.280
x5            -0.4893      0.257     -1.901      0.058      -0.995       0.017
x6             0.2945      0.209      1.406      0.160      -0.117       0.706
x7             0.0624      0.131      0.475      0.635      -0.196       0.320
x8             0.1094      0.100      1.097      0.273      -0.087       0.305
x9             0.4641      0.106      4.370      0.000       0.255       0.673
x10            0.0418      0.041      1.025      0.306      -0.038       0.122
==============================================================================
Omnibus:                        1.506   Durbin-Watson:                   2.029
Prob(Omnibus):                  0.471   Jarque-Bera (JB):                1.404
Skew:                           0.017   Prob(JB):                        0.496
Kurtosis:                       2.726   Cond. No.                         21.7
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
Parameters:  const   -1.804112e-16
x1      -6.184366e-03
x2      -1.481322e-01
x3       3.210963e-01
x4       2.003705e-01
```

```
x5      -4.893188e-01
x6       2.944779e-01
x7       6.241353e-02
x8       1.093696e-01
x9       4.640526e-01
x10      4.177106e-02
dtype: float64
```




```
Performance Metrics
Mean Squared Error: 0.4822505745867066
Mean Absolute Error: 0.5620021544511629
Manhattan distance: 248.40495226741407
Euclidean distance: 14.599820340241319
```

1.6 Common Training Parameters (GAN & ABC_GAN)

[7]:
```
n_epochs = 1000
error = 0.1
batch_size = 32
```

1.7 GAN Model

[8]:
```
real_dataset = dataset.CustomDataset(X,Y)
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
```

Training GAN for n_epochs number of epochs

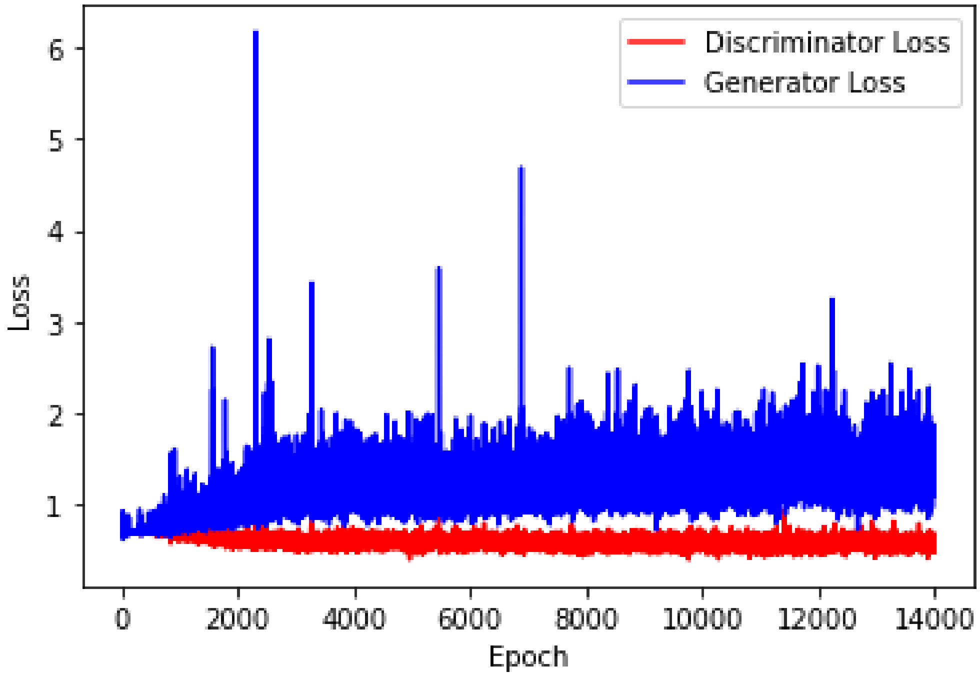
[9]:
```
generator = network.Generator(n_features+2)
discriminator = network.Discriminator(n_features+2)

criterion = torch.nn.BCEWithLogitsLoss()
gen_opt = torch.optim.Adam(generator.parameters(), lr=0.01, betas=(0.5, 0.999))
disc_opt = torch.optim.Adam(discriminator.parameters(), lr=0.01, betas=(0.5, 0.
 ↪999))
```

[10]:
```
print(generator)
print(discriminator)
```

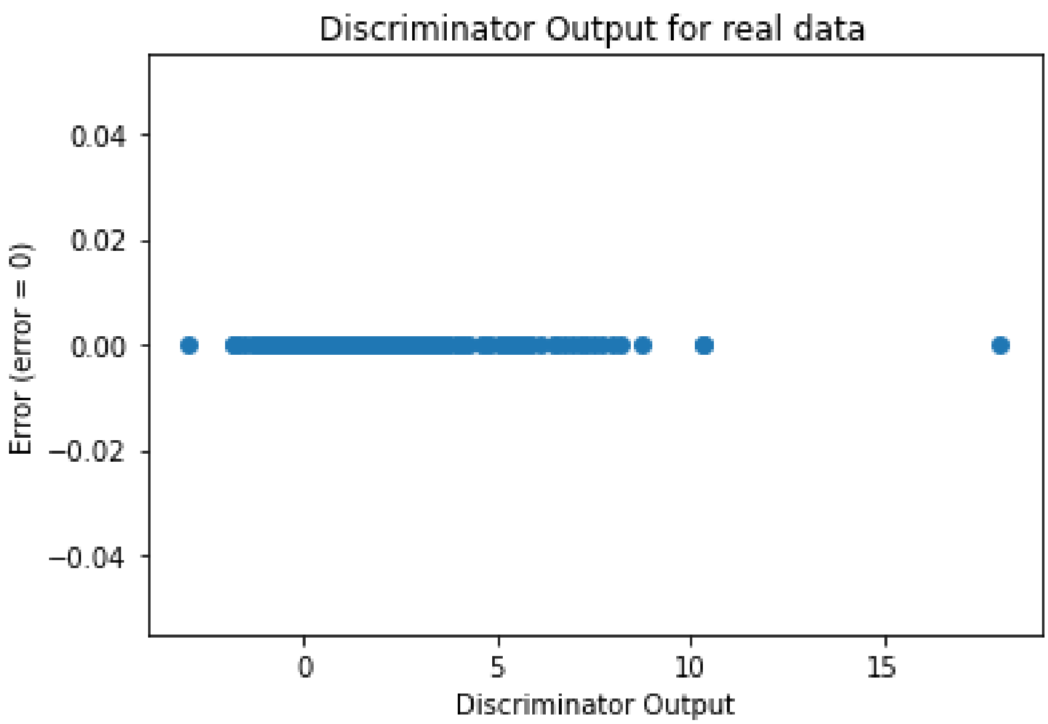
```
Generator(
  (hidden1): Linear(in_features=12, out_features=100, bias=True)
  (hidden2): Linear(in_features=100, out_features=100, bias=True)
  (output): Linear(in_features=100, out_features=1, bias=True)
  (relu): ReLU()
)
Discriminator(
  (hidden1): Linear(in_features=12, out_features=25, bias=True)
  (hidden2): Linear(in_features=25, out_features=50, bias=True)
  (output): Linear(in_features=50, out_features=1, bias=True)
  (relu): ReLU()
)
```

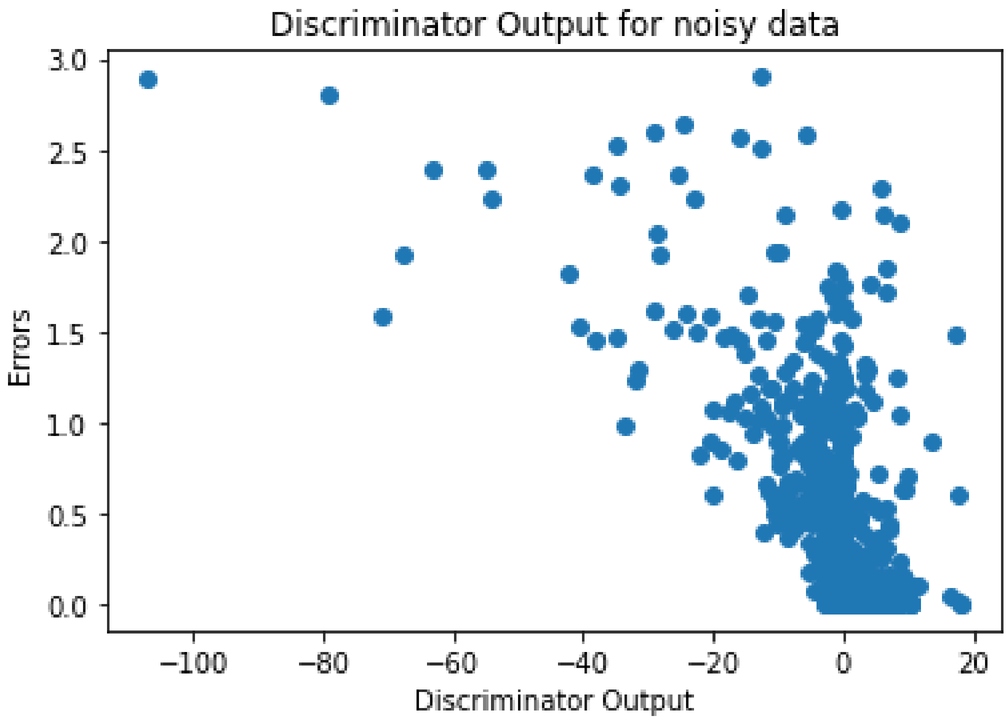
[11]:
```
train_test.
 ↪training_GAN(discriminator,generator,disc_opt,gen_opt,real_dataset,batch_size,␣
 ↪n_epochs,criterion,device)
```



[12]:
```
GAN1_metrics = train_test.test_generator(generator,real_dataset,device)
```

[13]:
```
sanityChecks.discProbVsError(real_dataset,discriminator,device)
```



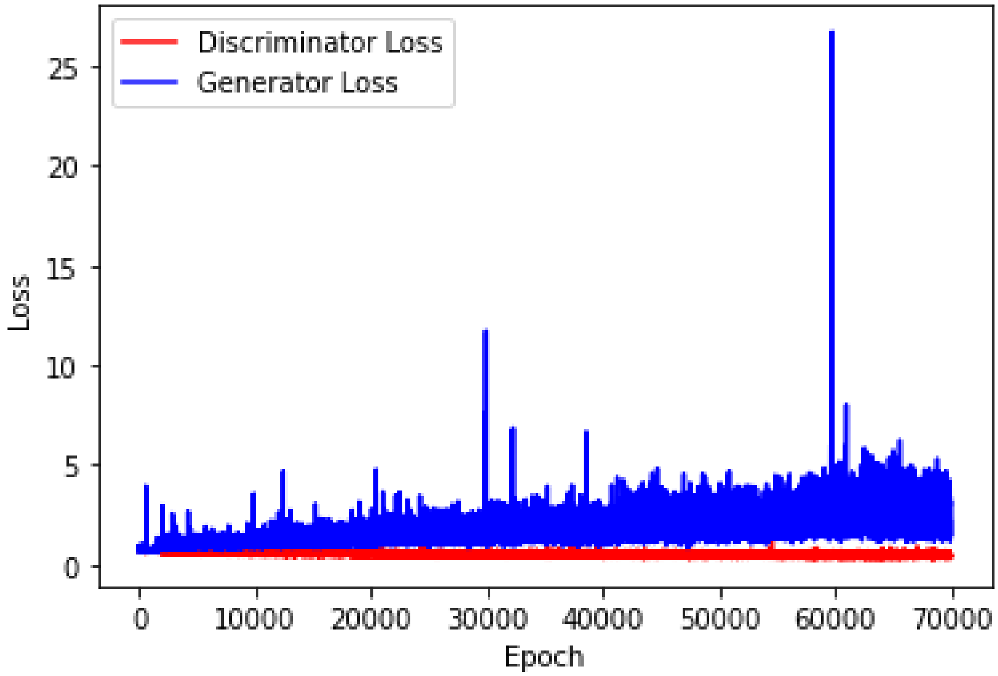


Training GAN until mse of y_pred is > 0.1 or n_epochs < 5000

[14]:
```
generator2 = network.Generator(n_features+2)
discriminator2 = network.Discriminator(n_features+2)
criterion = torch.nn.BCEWithLogitsLoss()
gen_opt = torch.optim.Adam(generator2.parameters(), lr=0.01, betas=(0.5, 0.999))
disc_opt = torch.optim.Adam(discriminator2.parameters(), lr=0.01, betas=(0.5, 0.
 ↪999))
```

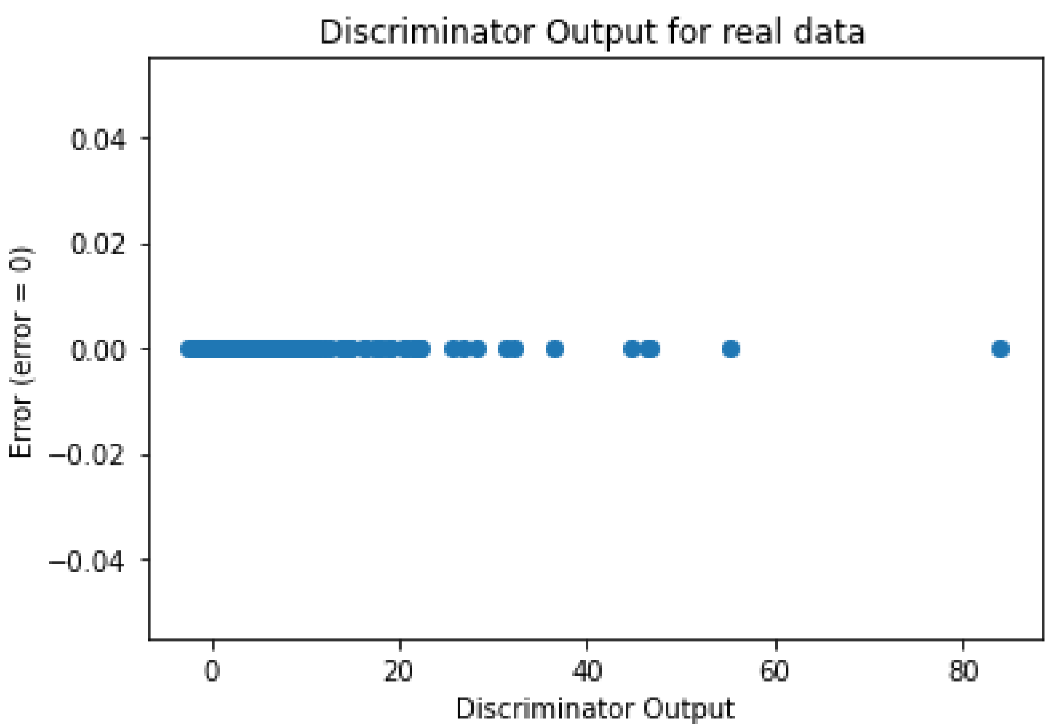
[15]:
```
train_test.
 ↪training_GAN_2(discriminator2,generator2,disc_opt,gen_opt,real_dataset,batch_size,error,cri
```

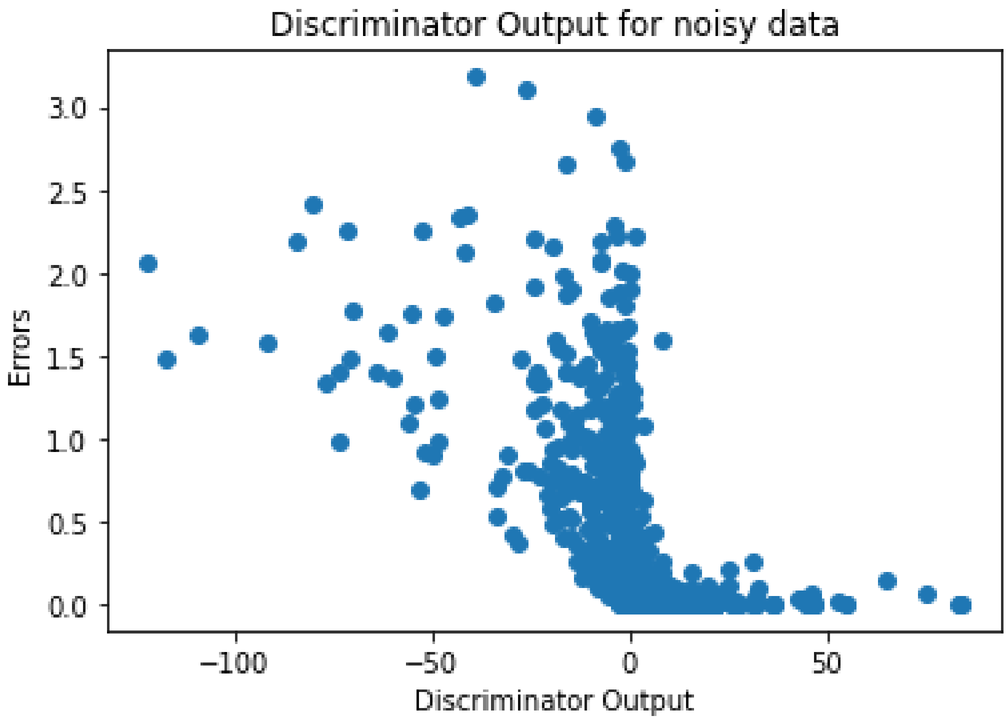
```
Number of epochs needed 5000
```



[16]:
```
GAN2_metrics=train_test.test_generator_2(generator2,real_dataset,device)
```

[17]:
```
sanityChecks.discProbVsError(real_dataset,discriminator2,device)
```





2 ABC GAN Model

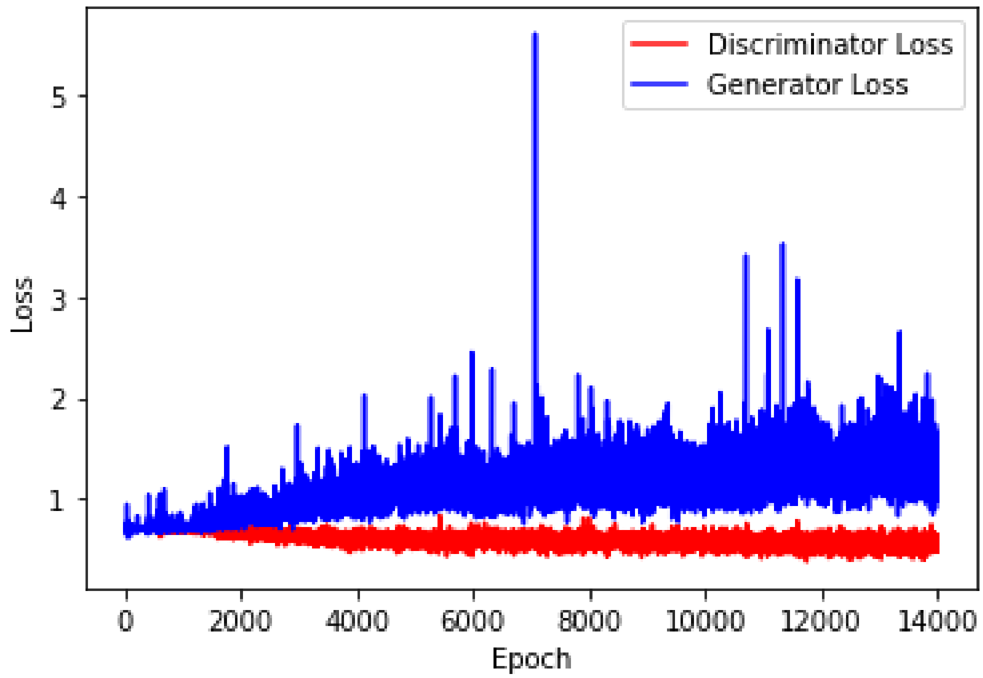
2.0.1 Training the network

Training ABC-GAN for n_epochs number of epochs

[18]:
```
gen = network.Generator(n_features+2)
disc = network.Discriminator(n_features+2)

criterion = torch.nn.BCEWithLogitsLoss()
gen_opt = torch.optim.Adam(gen.parameters(), lr=0.01, betas=(0.5, 0.999))
disc_opt = torch.optim.Adam(disc.parameters(), lr=0.01, betas=(0.5, 0.999))
```

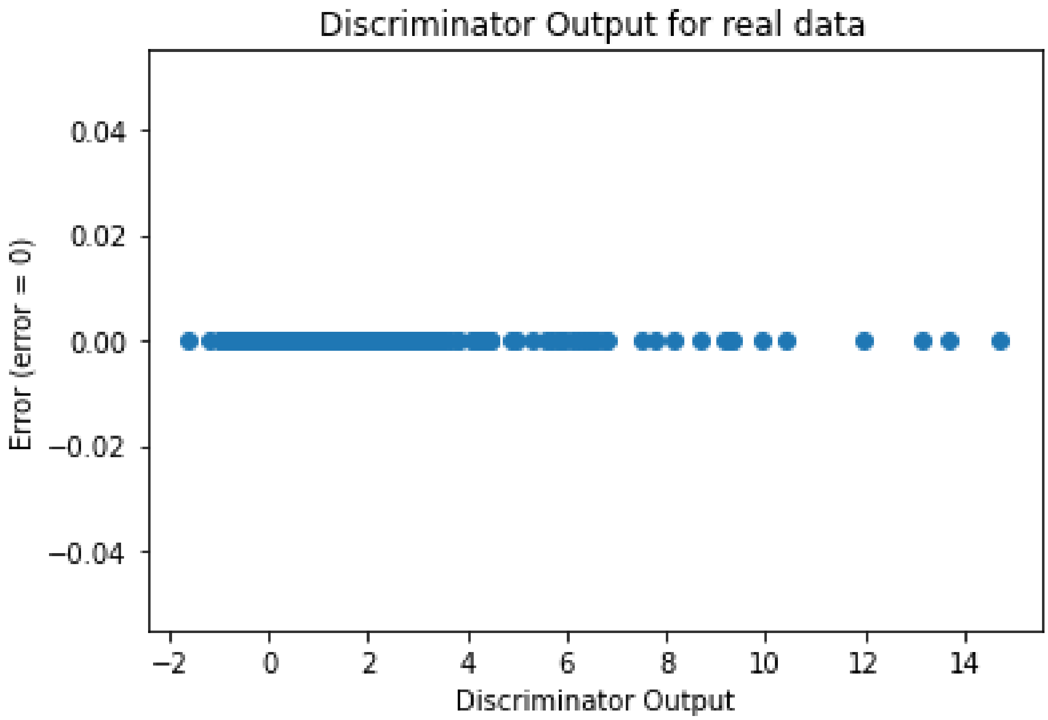
[19]:
```
ABC_train_test.training_GAN(disc, gen,disc_opt,gen_opt,real_dataset,␣
↪batch_size, n_epochs,criterion,coeff,mean,variance,device)
```

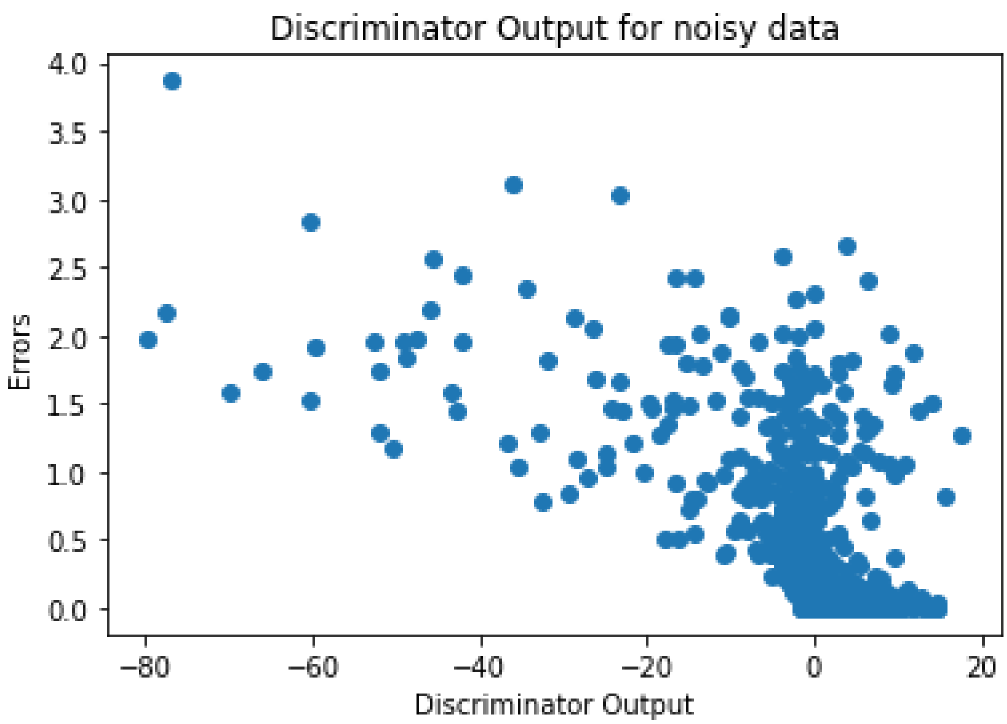


[20]:
```
ABC_GAN1_metrics=ABC_train_test.
↪test_generator(gen,real_dataset,coeff,mean,variance,device)
```

Sanity Checks

[21]:
```
sanityChecks.discProbVsError(real_dataset,disc,device)
```


Discriminator Output for real data
0.04
0.02
0.00
−0.02
−0.04
Error (error = 0)
−2
0
2
4
6
8
10
12
14
Discriminator Output


Discriminator Output for noisy data
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Errors
−80
−60
−40
−20
0
20
Discriminator Output

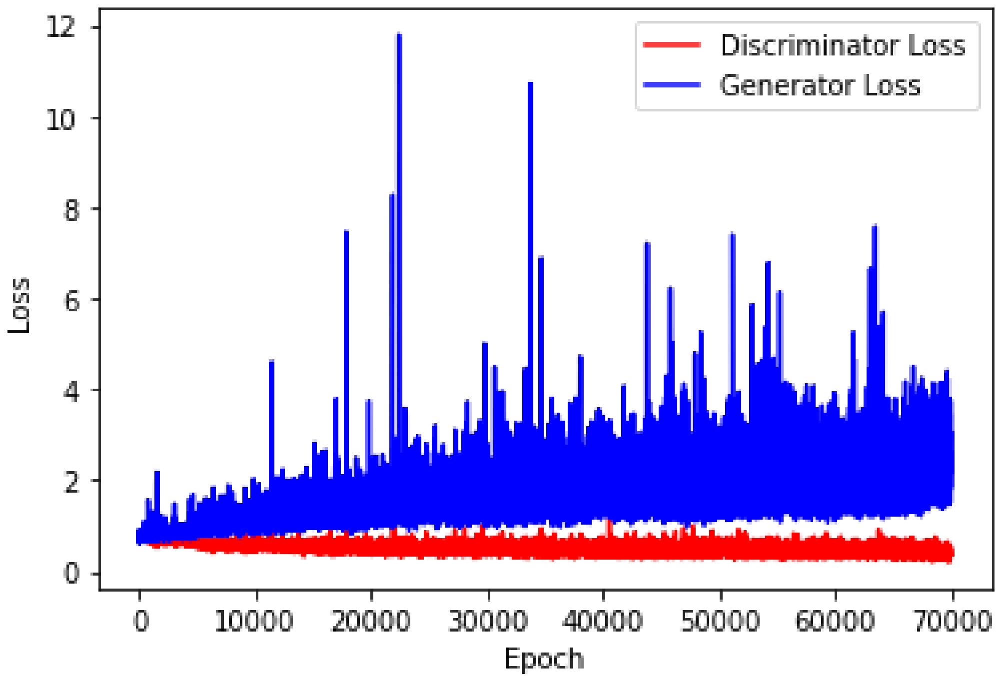
Training GAN until mse of y_pred is > 0.1 or n_epochs < 5000

[22]:
```
gen2 = network.Generator(n_features+2)
disc2 = network.Discriminator(n_features+2)

criterion = torch.nn.BCEWithLogitsLoss()
gen_opt = torch.optim.Adam(gen2.parameters(), lr=0.01, betas=(0.5, 0.999))
disc_opt = torch.optim.Adam(disc2.parameters(), lr=0.01, betas=(0.5, 0.999))
```

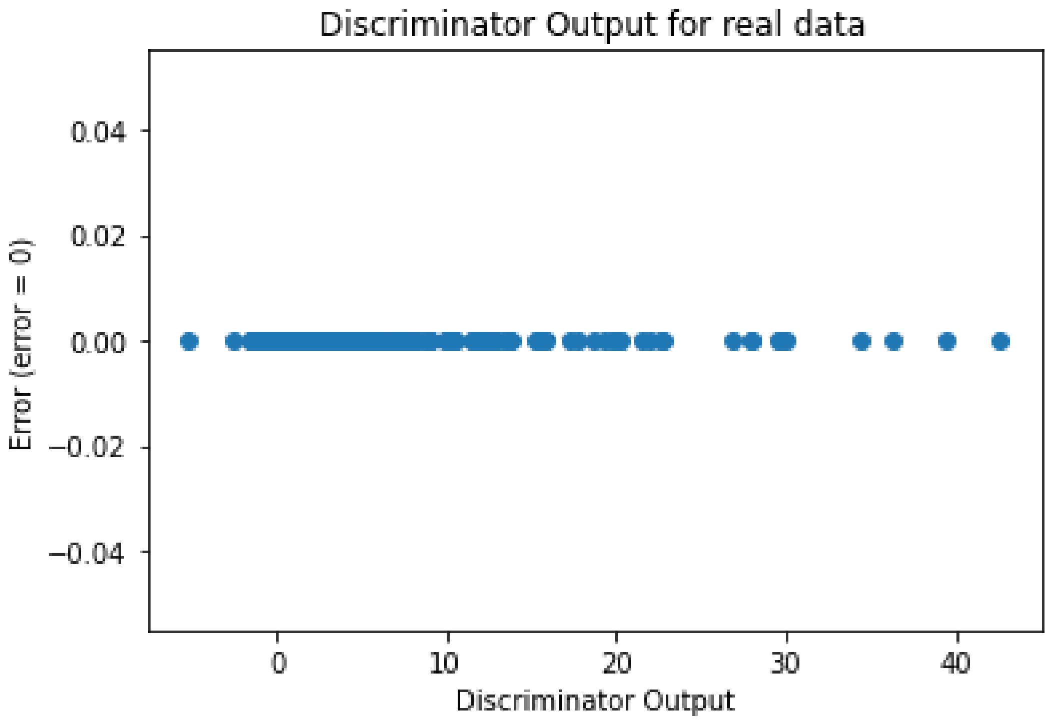
[23]:
```
ABC_train_test.
↪training_GAN_2(disc2,gen2,disc_opt,gen_opt,real_dataset,batch_size,␣
↪error,criterion,coeff,mean,variance,device)
```

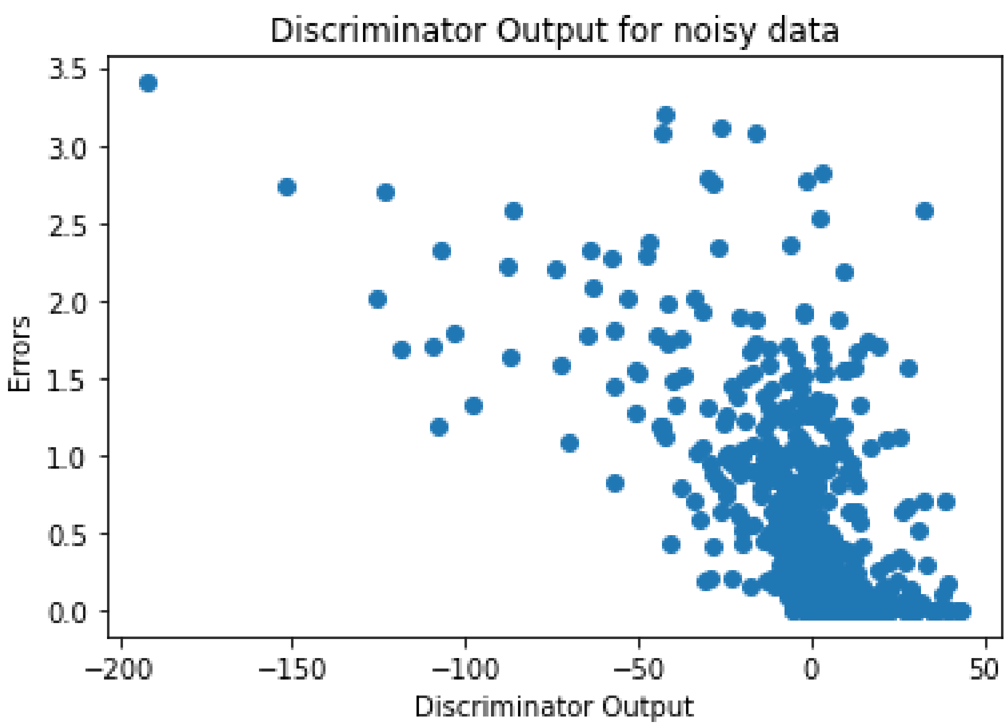
```
Number of epochs 5000
```



[24]:
```
ABC_GAN2_metrics=ABC_train_test.
↪test_generator_2(gen2,real_dataset,coeff,mean,variance,device)
```

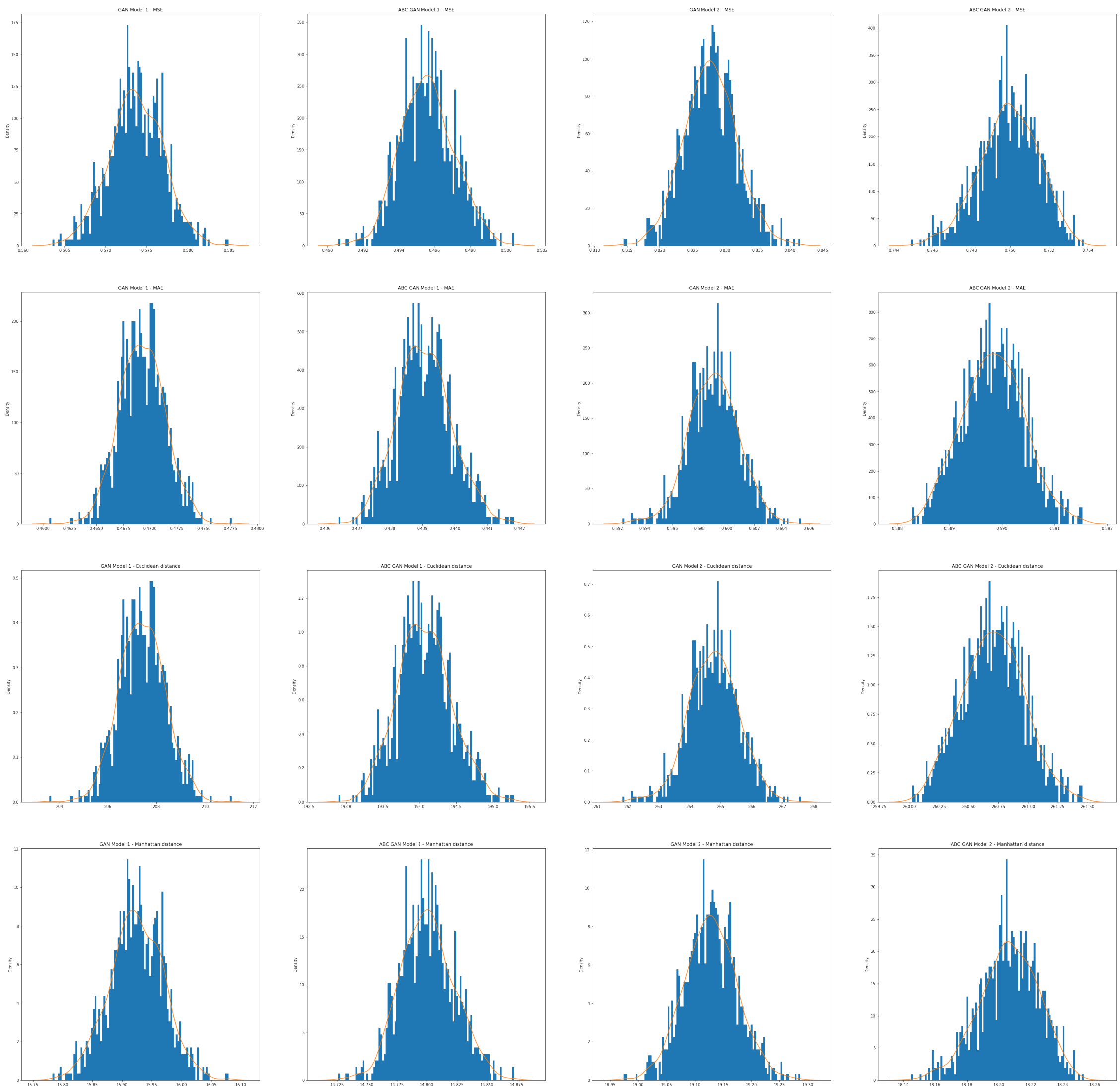
[25]:
```
sanityChecks.discProbVsError(real_dataset,disc2,device)
```


Discriminator Output for real data
0.04
0.02
0.00
−0.02
−0.04
Error (error = 0)
0
10
20
30
40
Discriminator Output


Discriminator Output for noisy data
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Errors
−200
−150
−100
−50
0
50
Discriminator Output

3 Model Analysis

[26]:
```
performanceMetrics.modelAnalysis(GAN1_metrics,ABC_GAN1_metrics,GAN2_metrics,ABC_GAN2_metrics)
```



[ ]:
```
```