First we describe the general GAN model that we created for both dataset, we note that the only change between the models is the input (noise dim) and output which is the desired sample size.

Generator:

Output  
[32, 9]

Dense  
[32, 40]

Dense  
[32, 20]

Dense  
[32, 10]

Input  
[32, 5]

Discriminator:

Output  
[32, 1]

Dense  
[32, 10]

Dense  
[32, 20]

Dense  
[32, 40]

Input  
[32, 9]

Dropout  
0.1

Dropout  
0.1

The table below shows the generator and discriminator input and output for each data set.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Dataset** | **Generator Input** | **Generator output** | **Discriminator Input** | **Discriminator output** |
| **Diabetes** | 5 | 9 | 9 | 1 |
| **German Credit** | 5 | 21 | 21 | 1 |

After the training process we give the generator random noise and receive an input data that is similar to the normalized data that we train the model on, that is not the intended output and thus we use the reverse\_transform method and receive a new dataset within the correct range of values.  
Following that, we let the discriminator guess if these samples are real or not and we label the discriminator guess for real samples as 1 and 0 otherwise. The example below shows the results for generating diabetes samples.

Table

Description automatically generated

In the following segments we will show further results and analysis on each data-set separately.

# *Diabetes*

First, we look at the diabetes dataset and preprocess it before continuing to the training process. We decided to use min-max normalization and we normalize the entire dataset with SKLearn function MinMaxScaler.  
The use of the MinMaxScaler with PCA resulted in a more easy-to-read plot where each group of sick/not sick points are divided clearly.  
Chart, scatter chart

Description automatically generated  
**Original data**

**Text

Description automatically generated**

**Generated data**

**Text, table

Description automatically generated with medium confidence**

**Learning process**

To capture and evaluate our GAN model we capture the weights of the network throughout a few milestones and then we generate some samples with those weights, comparing how close they are to the real data by closeness with PCA.

The results and progress of our model are clear and we see a great improvement as the epochs continues, the end result looks very close to the similar data except a few samples that sits somewhere between the two classes.

Chart, scatter chart

Description automatically generated Chart, scatter chart

Description automatically generated

When analyzing the actual samples after transforming them back to their original range of values we see similar behavior for most features, for example, plas and insu are quite close in values range for the original data, we can plot both data points at a 2D graph with the use of PCA and see how both groups distribute.  
The graph below shows the normalized data and the normalized generated data and we can see Chart, scatter chart

Description automatically generated

Overall, the samples are close to the original, but we do see an odd behavior at the left side of the plots with the line of dots, unfortunately we are not sure why this behavior is existed but that is the original data plotted with PCA so the dimension reduction might be the cause.

As of our discriminator, we want to check how many samples out of the 100 samples did fool the discriminator. We find out the 58 samples has been classified as true out of the 100 generated samples!

**Euclidean distance:**

We calculated the distance between positive examples on the real data and between the generated data to the real data and found out the following [Positive samples, Negative samples]:

Distance between real data points – Mean [160.30, 114.76]  
Distance between generated to real data points – Mean [185.30, 88.83]

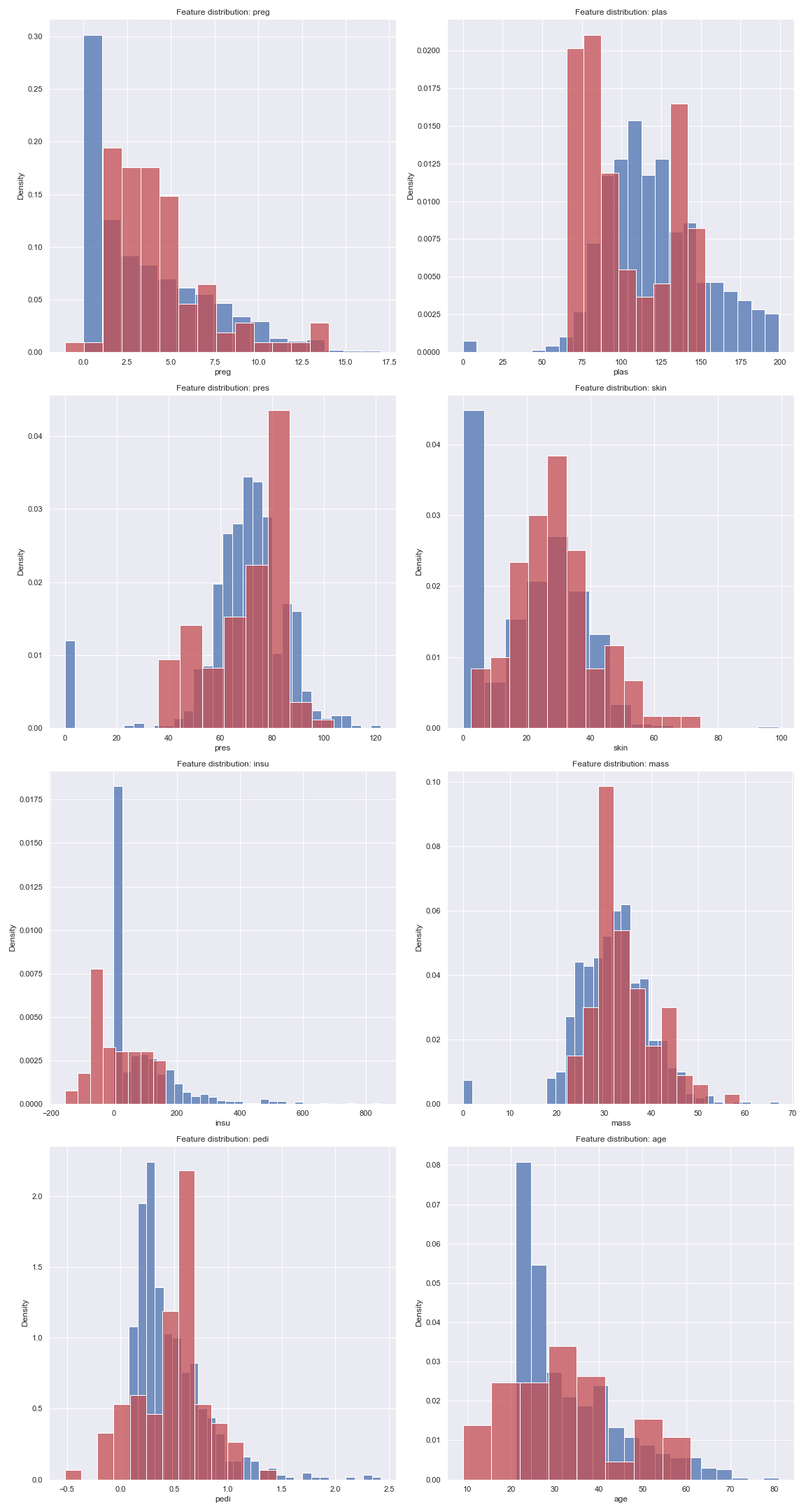
The results are very similar in terms of distance between points for the real data compared to the generated data.

**Loss graph:**

We do see a “back and forth” motion in the losses graph but it is not clear whether they switched places during this period, most of the times the clear “winner” was the discriminator and we think it is related to the ease of classification task in that particular dataset.

Chart

Description automatically generated



# German credit

As for the german dataset we used the StandardScaler method for normalization and the results were:  
Table

Description automatically generated

**Original data**

**Table

Description automatically generated**

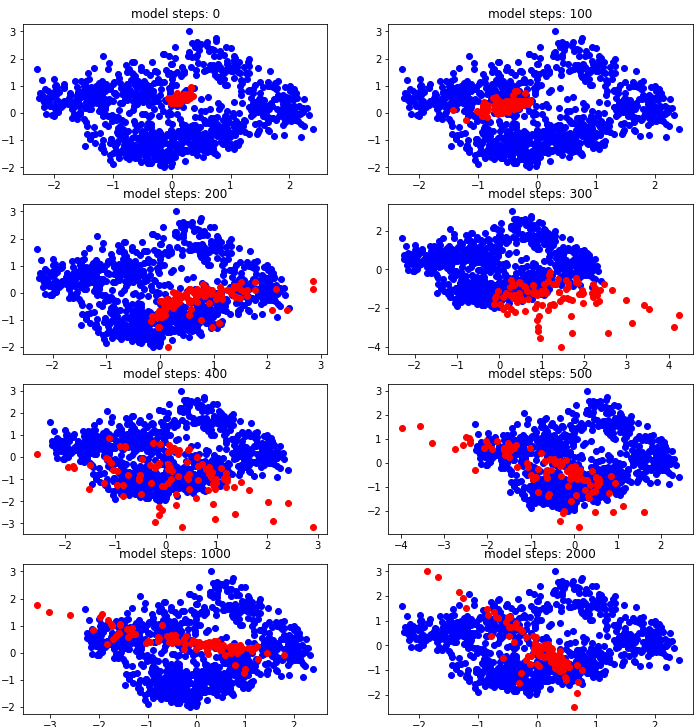
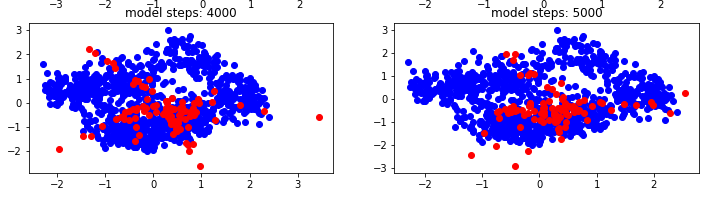
**Generated data**

**Table

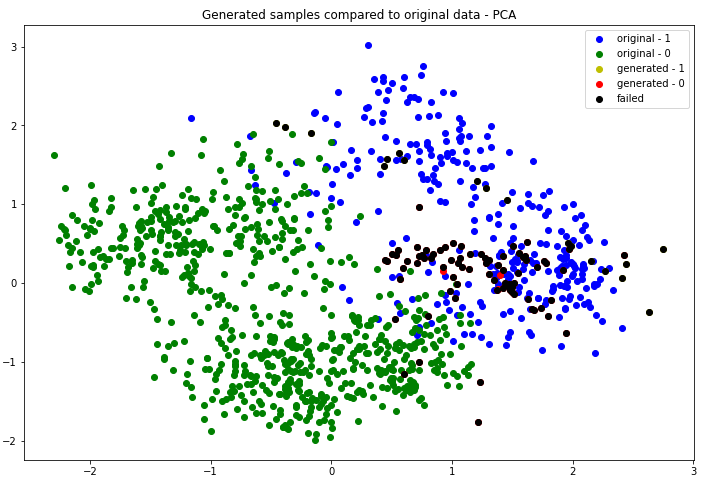
Description automatically generated**

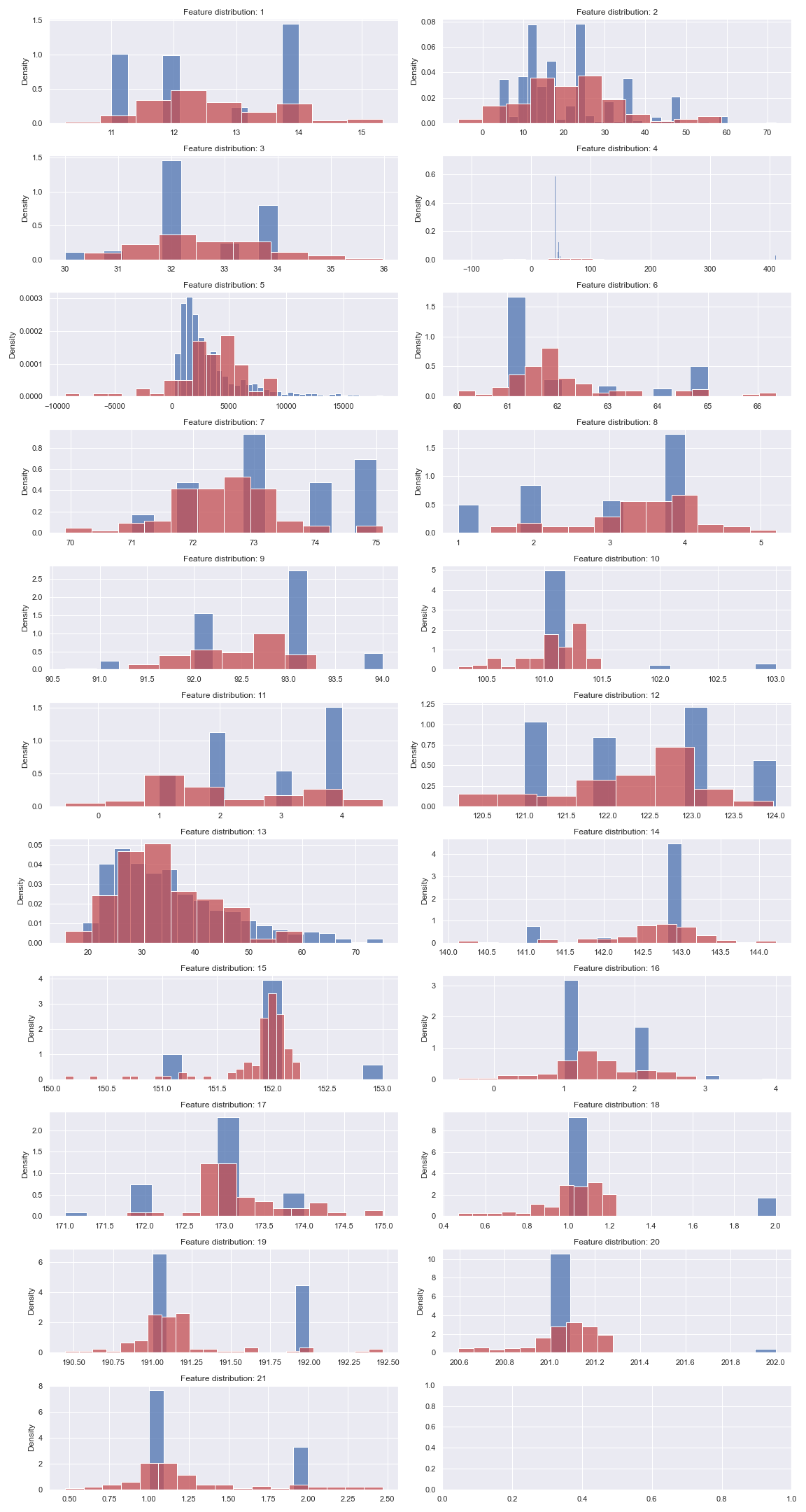
**Learning process**

The process of generating samples in the German credit was considerably harder we assume because of facts that the dataset contains a lot more features that needs to be generated and correlated to each other and the unbalance nature of the data-set.

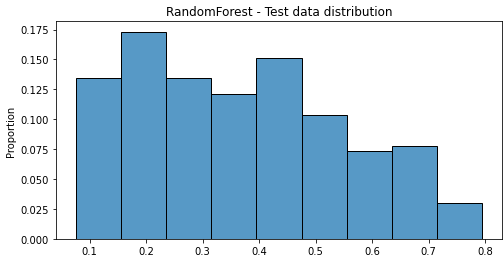
   
Similar to the diabetes data we attempt to create a PCA representation of the generated and original data to conduct a direct comparison between the two and see if the samples are close to each other.

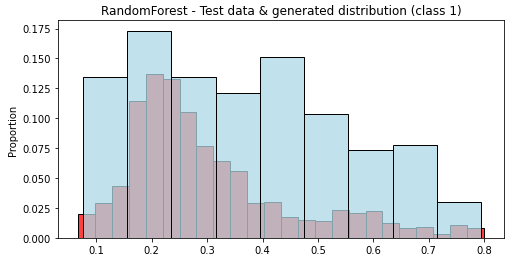
But as we can se below, unlike the diabetes data, this data set is a bit more complicated and the clear distinction between both classes is not well defined. We do see that generated samples that tends to be more “clear” and far from the middle which is the vague part of samples are easier to classify but most data that lays in the center is harder for the discriminator to classify.



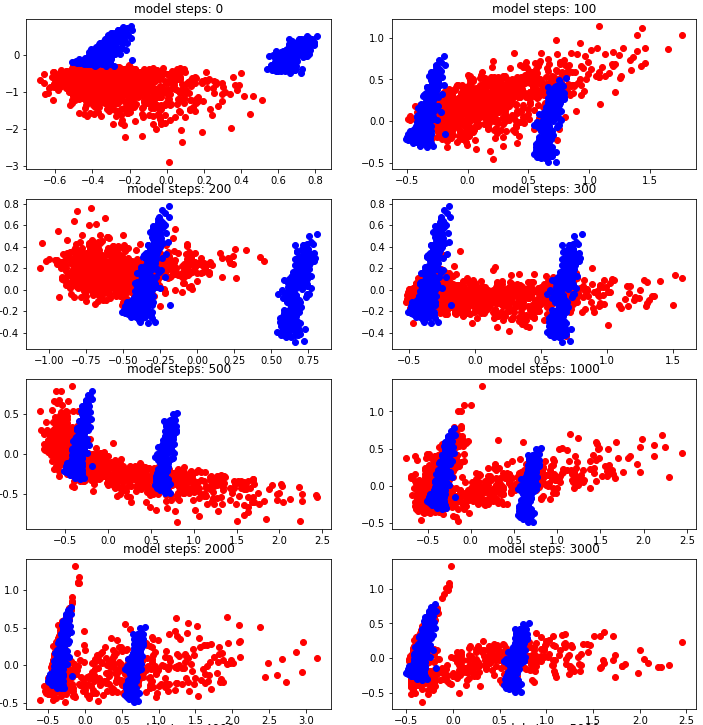


**PART 2**



Chart, histogram

Description automatically generated

Chart, scatter chart

Description automatically generated