**Report**

Assignment 4, Deep Learning

by Carmel Shablin 305812661

and Gregory Koushnir 321889479

## Task

Implement two tabular data generative architectures: standard GAN, and a modified GAN architecture, which is inferring the inner-worker black box (BB) model.

## Dataset

The datasets include 2 files of taular data

a. German\_credit.arff – credit card samples with categorical and numeric features, the target is a good/bad customer.

b. diabetes.arff – for blood measurements and general pregnancy information, target is a patient has diabetes or not.

## Environment

Since none of the students has or has access to GPU physically, the assignment was done in the Google Colab environment in order to speed up the training process. All shown results are taken from the Jupiter notebook.

## Method

The general method is to build a GAN for each dataset, GAN is combined from two NN: one a Generator and the other is Discriminator. The main goal is to make Generator generate samples that are similar to the real ones.

### Preprocessing:

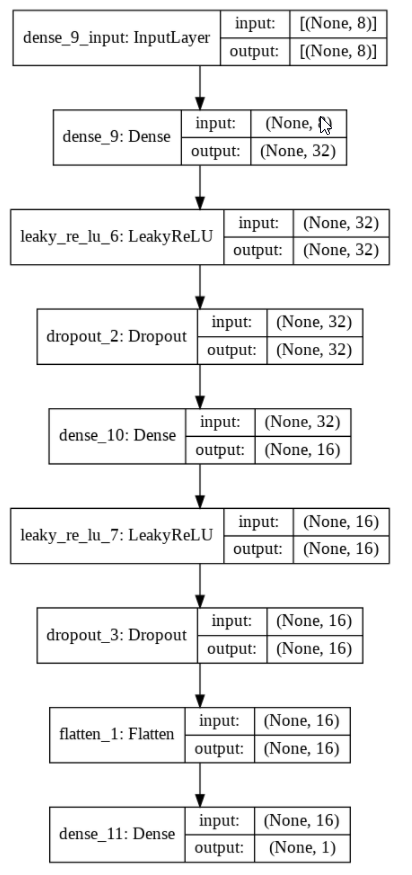
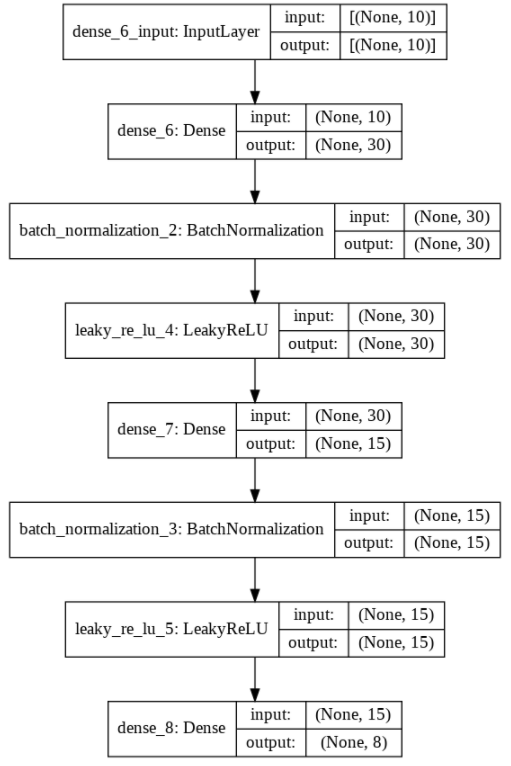
diabetes.arff – all columns were numeric so we just min-max scaled all columns to [-1, 1]

German\_credit.arff – some of columns are categorical so for those we performed one-hot-encoding. Than, all of the columns are min-max scaled to [-1, 1]

### Architecture

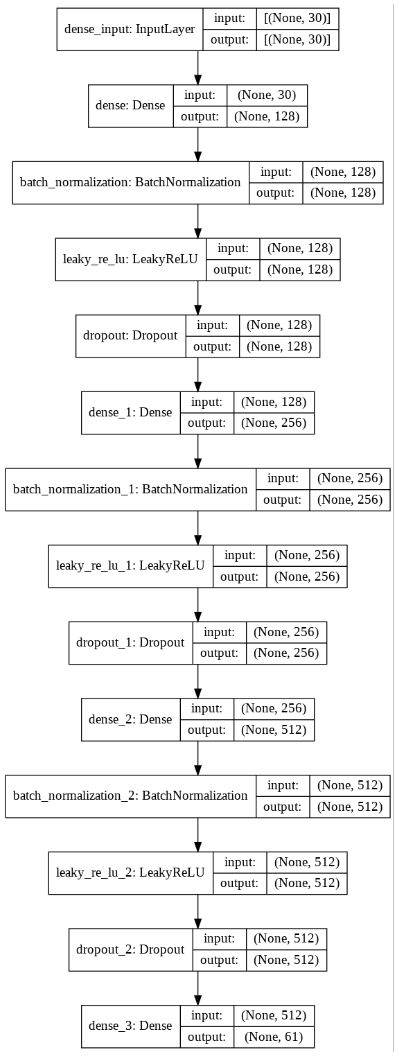
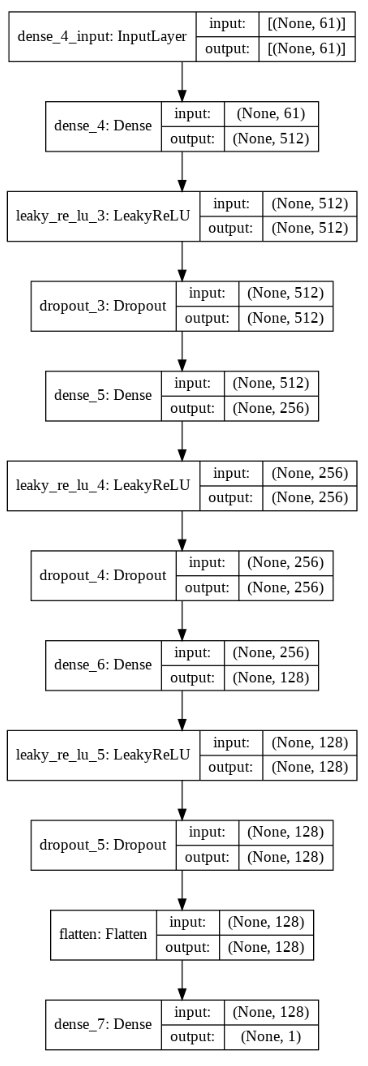
Similar architecture was implemented (per dataset) for both parts of the assignment. The architecture was chosen empirically and proved to get the best results compared to others.

Diabetic GAN, Generator + Discriminator respectively:



The Generator has two blocks of Dense-BatchNormalization-LeakyReLU between the input and output layers. The output activation function is a hyperbolic tangent, which fits the preprocessing features scaling. The Discriminator has two blocks of Dense-LeakyReLU-Dropout (rate=0.2) between the input and output layers. The D loss is binary-crossentropy on the sigmoid activation of the output neuron. Adam optimizer was used for both the discriminator (learning rate = 2e-4, =0.5) and the combined GAN (learning rate = 2e-4, =0.5) architectures. For the training, equal number of stochastically selected samples from the dataset with target=1 and randomly generated samples from the latent noise (uniformly distributed) vector with dimension of 10 and target=0 were fed to the discriminator to learn to determine between real and generated (fake) samples.

Credit GAN, Generator + Discriminator respectively:

The architecture of the GAN is almost the same as for Diabetic GAN with the following differences:

Dropout layers were added in the Generator (rate=0.25). Generator has three Dense-BatchNormalization-LeakyReLU-Dropout blocks instead of two. Discriminator has three Dense-LeakyReLU-Dropout (rate=0.2) blocks instead of two. Adam optimizer parameters for both the discriminator (learning rate = 1.5e-5, =0.95) and the combined GAN (learning rate = 2e-4, =0.6, =0.999) architectures were changed. The noise latent vector size was changed to 30.

In addition, the number of the hidden units was changed as might be seen from the architecture description in a figure above.

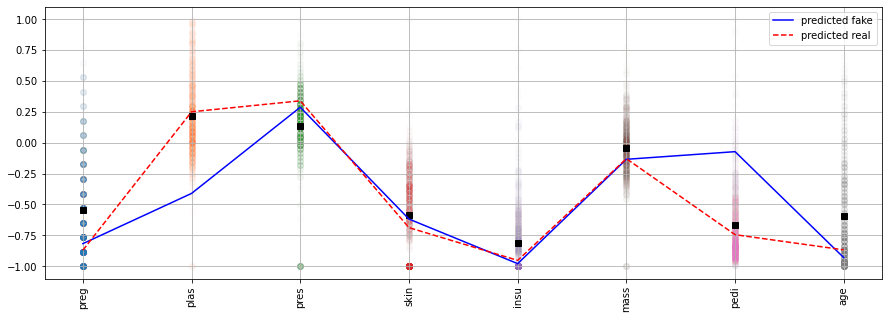
### Part 1

#### Diabetic dataset

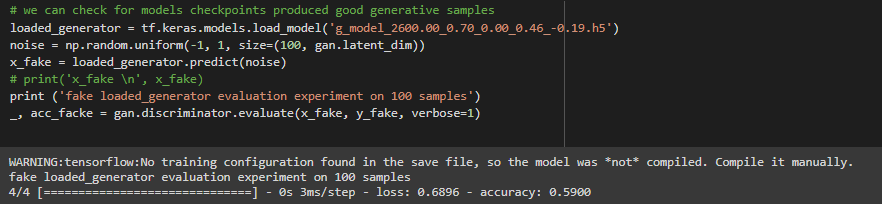
1. NA
2. GAN convergence
   1. GAN losses reached convergence as can be seen in plot of ‘3.c’

To achieve the best performance model, each 50 iterations after the first 800 iterations, we check for convergence by measuring the difference of the loss of generator and discriminator and requiring it to be less than 0.2. After that, we check that the discriminator loss is in 0.2 tunnel from 0.5 (the optimal discriminator mistakes in half of samples) and also that the discriminator loss on real and fake images is nearly same (d\_fake\_accuracy is at most 0.2 from the d\_real\_accuracy). We save every model that applies the constraint. The very first model was achieved after 2600 iterations. Both G and D losses were 0.7, D accuracy was 0.46 and the difference between the D’s accuracies for fake and real samples was 0.19.

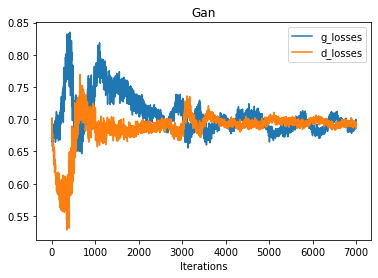
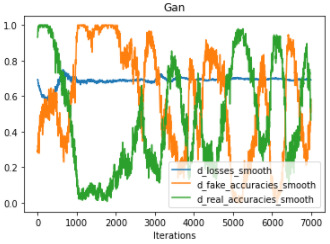
1. Analysis



* 1. From the above plot, we may see the feature distribution and the mean value of each feature marked by a black square. For the second and fifth features, the sample that was predicted as a real one the values are much closer to the mean than for the sample that was evaluated as a fake by the discriminator.
  2. Out of 100 samples 59 were marked as “fake” and 41 as “real”:



* 1. As can be seen the models did go back and forth there was no consistent leader:

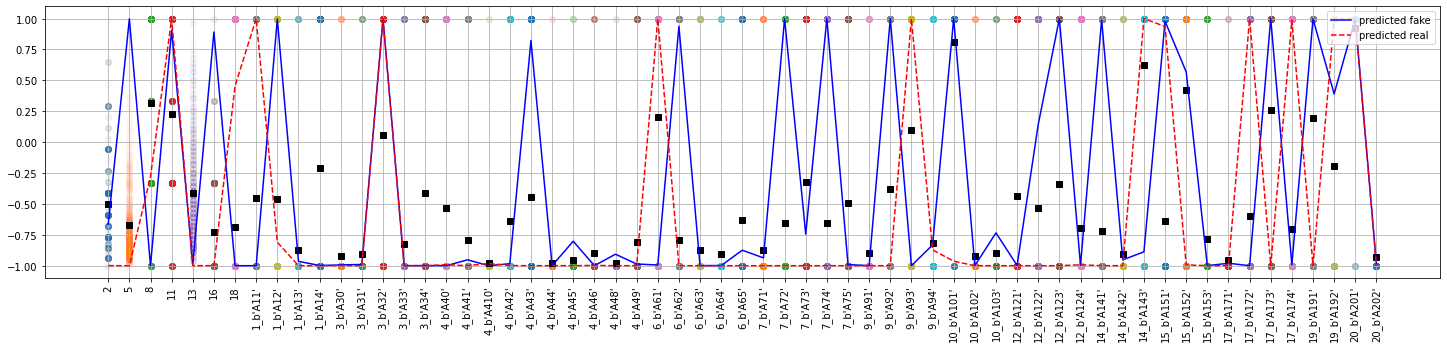
 

#### Credit dataset

1. NA
2. GAN convergence
   1. GAN losses reached convergence as can be seen in plot of ‘3.c’

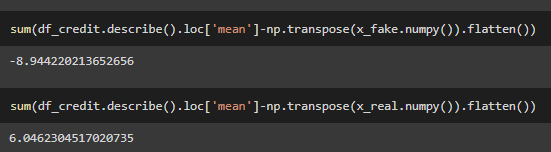
The same criteria for a best model selection as used for the diabetic dataset was used.

1. Analysis

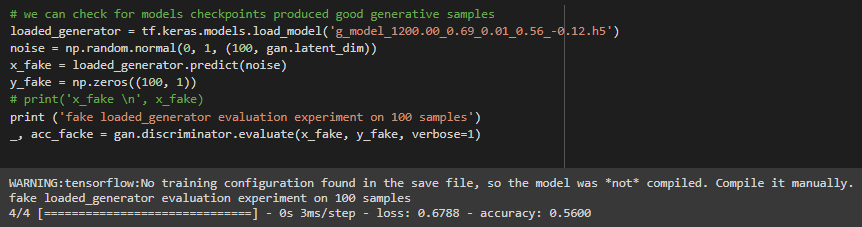


* 1. Because of the binary nature of categorical one-hot encoding, the total amount of features is much greater than in previous case and the distribution is strict {-1,1} (after the preprocessing). Statistically we may see the number of mean values above zero and count the misses by “real” and “fake” samples’ values. In a summary, the count of predicted “fake” misses is greater than the number of predicted “real” misses, which make sense to us.

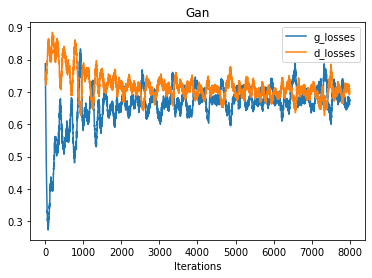
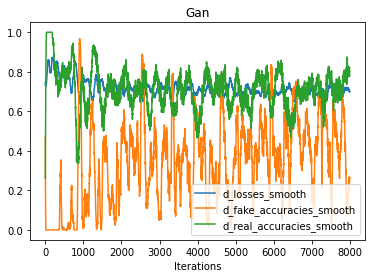
The conclusion may be also examined by measuring the Euclidian distance between the mean and “fake”/”real” sample vectors. The distance from mean to real is smaller than the distance from mean to fake:



* 1. Out of 100 samples 56 were marked as “fake” and 44 as “real”:



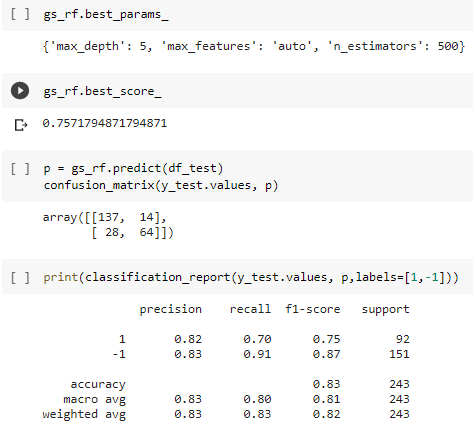
* 1. As can be seen the models did go back and forth and that the “real” prediction was preferable by the discriminator:

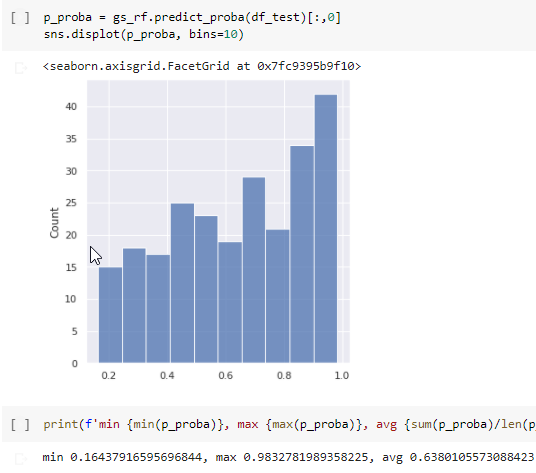
### Part 2

#### Diabetic dataset

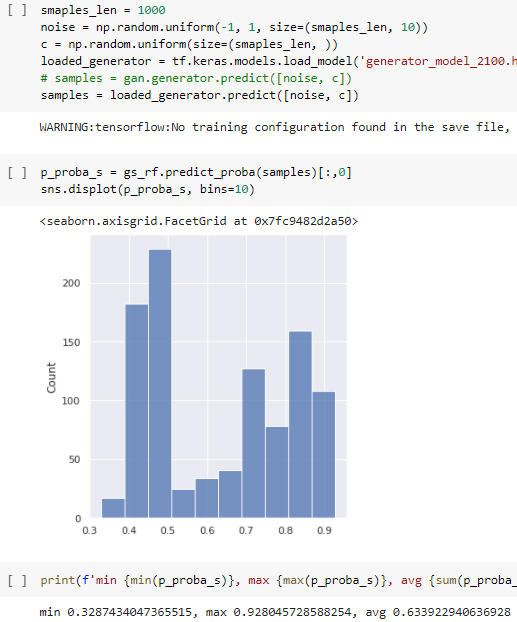
1. Random forest trained (on 70%) using 5-fold CV and Grid search and the following found to be the best parameters:



1. NA
2. NA
3. Following information provided:
   1. Random Forest test (30%) performance:



* 1. NA
  2. Statistics on score distribution of BB using generated samples:



Looks like the resulting distribution is Uniform-alike, boundaries are wide almost as in real distribution (much better than in the other dataset GAN). The perfect expectation would have been close to ~ U(0,1) same as the distribution of sampled “Confidences”.

TODO answer 2 q:

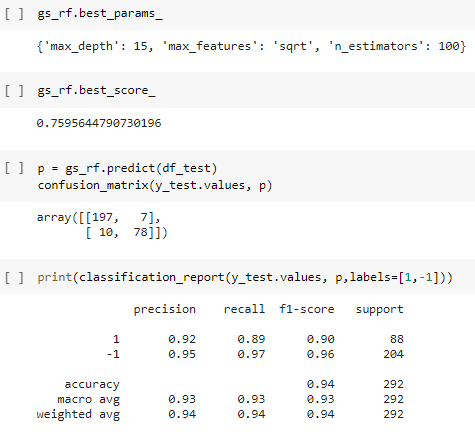
Were you more successful for a specific class of samples? 🡺 means low confidence is class\_1 and high is class\_2, so we can test/plot distribution for both classes

Were you more successful for a specific range of confidence scores? 🡺 to sample from deferent ranges “Confidences” and check the output range, desired expectation would be c = Y i.e. the resulting range will be same

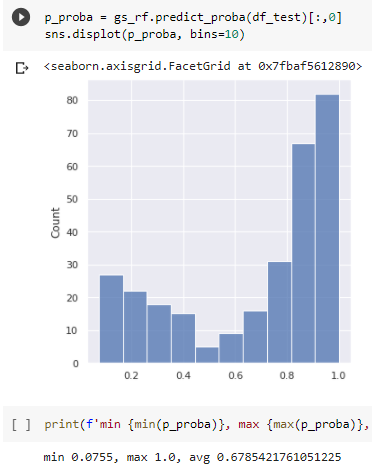
* 1. problematic aspect we had encountered influenced the resulting ‘Confidence’ distribution - Firstly our ‘noise’ for generation was sampled from ~ N(0, 1) and the BB output distribution was worse from now (final state) it was always Normal-alike narrow distribution, we expected it to be more Uniform alike with boundaries close to [0, 1]. So what we did is to sample ‘noise’ from ~ U[-1, 1] this improved greatly the BB output distribution that looked unformal-alike~ U[a, b] as can be seen in ‘4.c’ plot.

#### Credit dataset

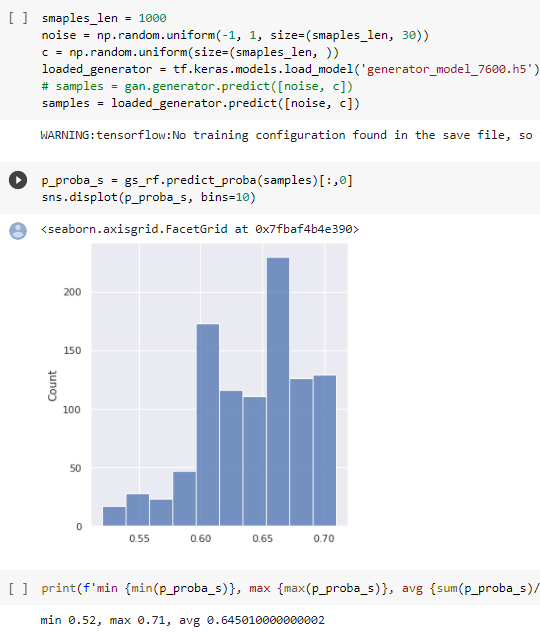
1. Random forest trained (on 70%) using 5-fold CV and Grid search and the following found to be the best parameters:



1. NA
2. NA
3. Following information provided:
   1. Random Forest test (30%) performance:



* 1. NA
  2. Statistics on score distribution of BB using generated samples:



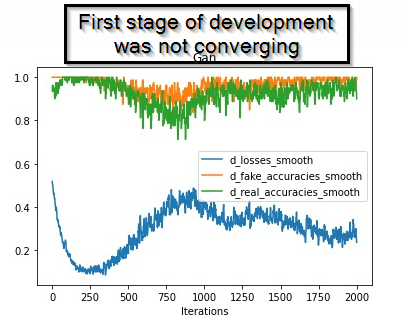
Looks like the resulting distribution is lacking samples near the boundaries (0 and 1) the perfect expectation would have been ~ U(0,1) same as the distribution of sampled “Confidences”.

TODO answer 2 q:

Were you more successful for a specific class of samples? 🡺 means low confidence is class\_1 and high is class\_2, so we can test/plot distribution for both classes

Were you more successful for a specific range of confidence scores? 🡺 to sample from deferent ranges “Confidences” and check the output range, desired expectation would be c = Y i.e. the resulting range will be same

* 1. At Early stages of development, the model suffered from not converging – the Discriminator was much better than Generator, as showed below in plot the discriminator can tell what is real and fake and the desire is to fool it by 50% of samples:



So we had to make them converge / learn in equal rate, we slowed down ‘D’ by changing learning rate by x10 to 0.0002 and later fine-tuned ‘D’ beta\_1🡺0.8, ‘G’ beta\_1🡺0.6 .

Another problematic aspect we had encountered influenced the resulting ‘Confidence’ distribution - Firstly our ‘noise’ for generation was sampled from ~ N(0, 1) and the BB output distribution was worse from now (final state) it was always Normal-alike narrow distribution, we expected it to be more Uniform alike with boundaries close to [0, 1]. So what we did is to sample ‘noise’ from ~ U[-1, 1]. This step improved greatly the BB output distribution that looked uniformal-alike~ U[a, b] as can be seen in ‘4.c’ plot.