Deep Learning KU (708.220) WS22

Assignment 2: Neural Networks: Regression

David Mihola
12211951

Massimiliano Viola
12213195

November 30th, 2022

Contents

a) Selection and creation of a data sets

We chose to use the `social_capital_zip.csv` data set, for this assignment. Firstly, we made a simple analysis of the completeness of the data and noticed that a significant portion of the rows contained missing values. In particular, out of the 23028 total rows, only 14269 were such that all variables were not NaN.

Having already decided that the target variable would be `ec_zip`, explained in **b)**, we decided to remove all rows where it was NaN as part of the pre-processing. We thought that imputing these missing values and then predicting them would not make sense.

On the other hand, we filled the missing values of each of the other chosen explanatory variables, listed in **b)**, with the mean of the corresponding column. We thought that mean imputation was appropriate because the domains of these variables seem to be continuous.

Secondly, although normalization was not strictly necessary, i.e the values of all variables were quite small and not far from zero, we normalized the variables used for prediction just for good practice to have 0 mean and standard deviation of 1.

b) Design of a regression problem

As a regression problem, we decided to predict the value of the main variable of the data set, that is the `ec_zip` variable. This variable contains values of economic connectedness, i.e. *The extent to which people with different characteristics (e.g., low vs. high socioeconomic status) are friends with each other* as described in [1], among people living in the area with the same ZIP code. The `ec_zip` variable is dimensionless.

As the input for our model we chose the 10 most positively or negatively correlated variables of this data set with the target variable. These variables turn out to be, sorted by absolute correlation: `ec_grp_mem_zip`, `exposure_grp_mem_zip`, `nbhd_ec_zip`, `ec_high_zip`, `nbhd_exposure_zip`, `ec_grp_mem_high_zip`, `exposure_grp_mem_high_zip`, `nbhd_ec_high_zip`, `nbhd_bias_high_zip`, `bias_grp_mem_zip`.

All these variables are continuous while other variables, such as `zip` or `county`, are of no interest as they are basically unique categorical identifiers, although they are numerical.

c) Neural network architecture selection

We started by defining a simple linear regression baseline model using the ten selected variables. This simplest possible model allows us to compare the performance of other neural networks, to see whether they add any value and if so, by how big of a margin. The performance of the baseline linear model can be seen below in figure 1 with a baseline MSE of around 0.002 for both training and validation data. By inspecting the coefficients, it can be seen in figure 2 that basically only three variables contribute a lot to the predictions.

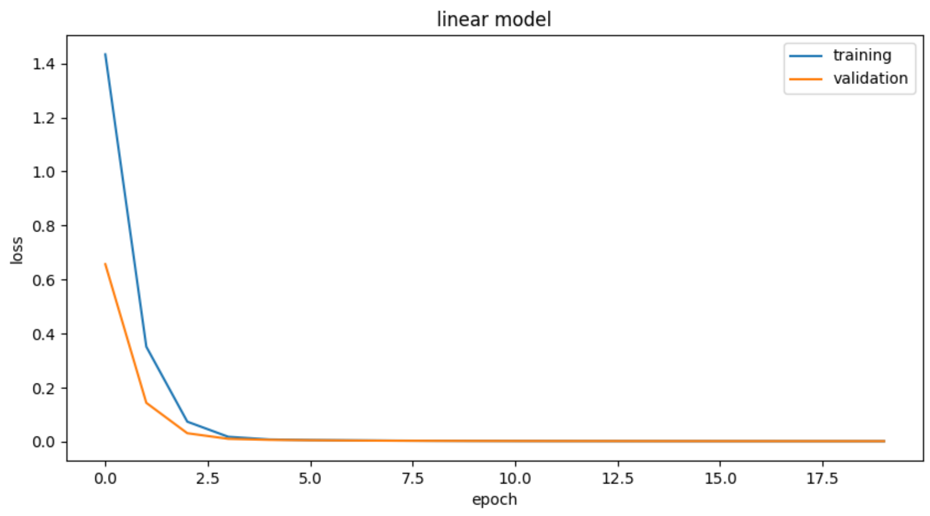


Figure 1: Validation and training loss of the linear model

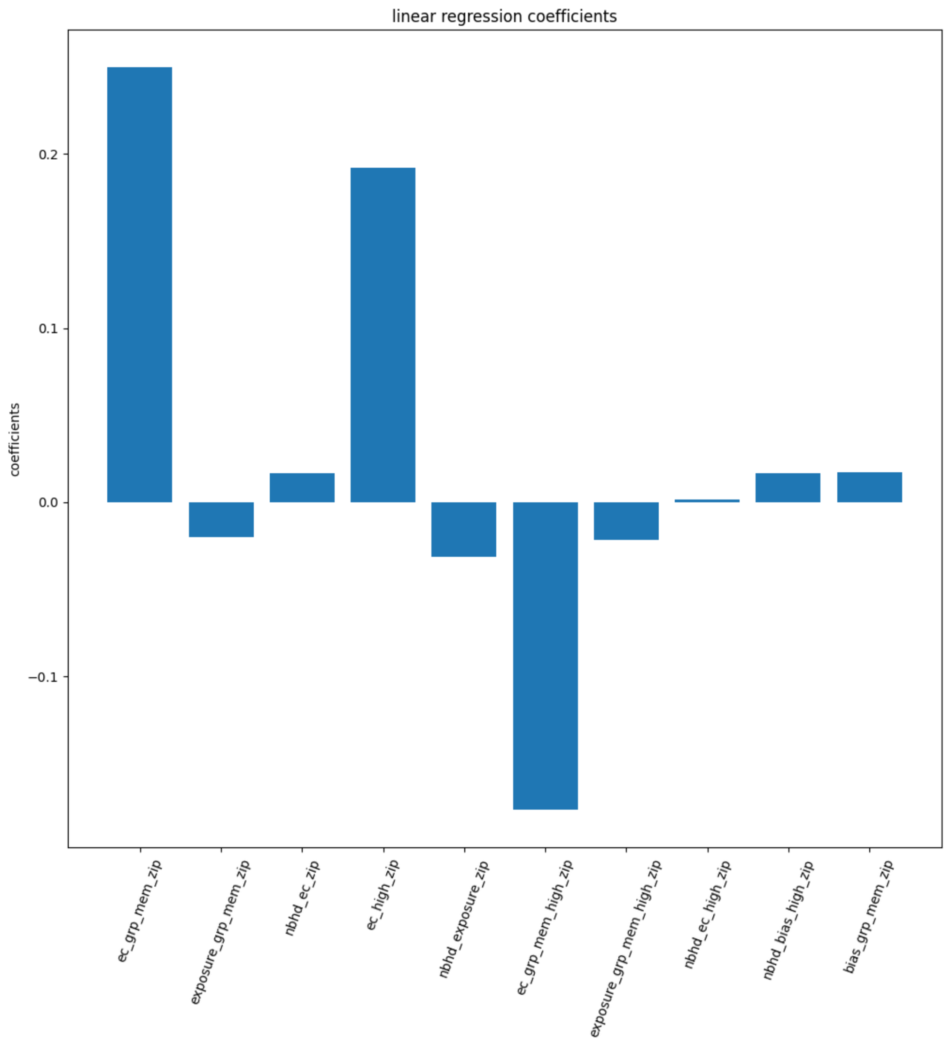


Figure 2: Coefficients of the linear model

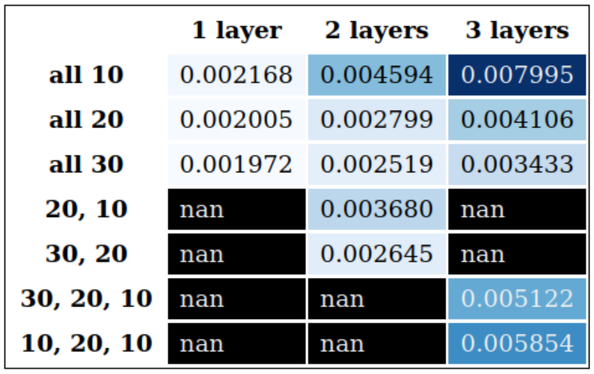
For our experiments, we thought that a neural network (NN) with a couple of hidden layers of reasonably small size should perform the best, as the chosen regression problem is not very complicated and we don't want to overfit. Therefore, we tested NNs with 1 to 3 hidden layers and 10 to 30 units per layer. We also added dropout layers to regularize with a factor 0.4 after each hidden layer excluding the last hidden layer. We used the default settings of the Adam optimizer and a batch size of 32 samples. We trained the NNs for 20 epochs as the loss stabilized in most cases. The results in form of errors for the training data and errors for the validation data can be seen below in tables in figure 3.

The table columns specify the number of hidden layers, while the rows specify the number of units/neurons in each hidden layer. Cells with *nan* represent, that a NN with this configuration could not be built.

It is evident from the results that the simpler NNs just with one hidden layer outperform the more complicated ones with 2 or 3 hidden layers. We conjecture this happens because they are very complex models and overfit the data in these simple settings. This is supported by the fact that the gap between training and validation loss increases with the number of hidden layers.

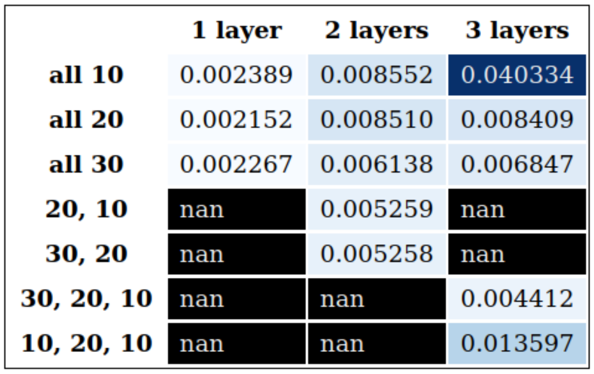
We observe also that the training and validation losses for the simpler NNs are basically the same of the ones from linear model baseline, meaning adding parameters and degrees of freedom does not help much.

The best NN from the experiments above is the NN with one hidden layer consisting of 20 units. We used this NN in **c)** and **d)**.



	1 layer	**2 layers**	**3 layers**
all 10	0.002168	0.004594	0.007995
all 20	0.002005	0.002799	0.004106
all 30	0.001972	0.002519	0.003433
20, 10	nan	0.003680	nan
30, 20	nan	0.002645	nan
30, 20, 10	nan	nan	0.005122
10, 20, 10	nan	nan	0.005854

(a) Loss on the training data



	1 layer	**2 layers**	**3 layers**
all 10	0.002389	0.008552	0.040334
all 20	0.002152	0.008510	0.008409
all 30	0.002267	0.006138	0.006847
20, 10	nan	0.005259	nan
30, 20	nan	0.005258	nan
30, 20, 10	nan	nan	0.004412
10, 20, 10	nan	nan	0.013597

(b) Loss on the validation data

Figure 3: Losses after 20 epochs

To see how the models behaved during training, we also plotted the training and validation losses for 4 of them in figure 4. In this figure, it is even more evident, how not even the most complicated models overfit the training data as the validation loss starts to rise instead of following the training curve.

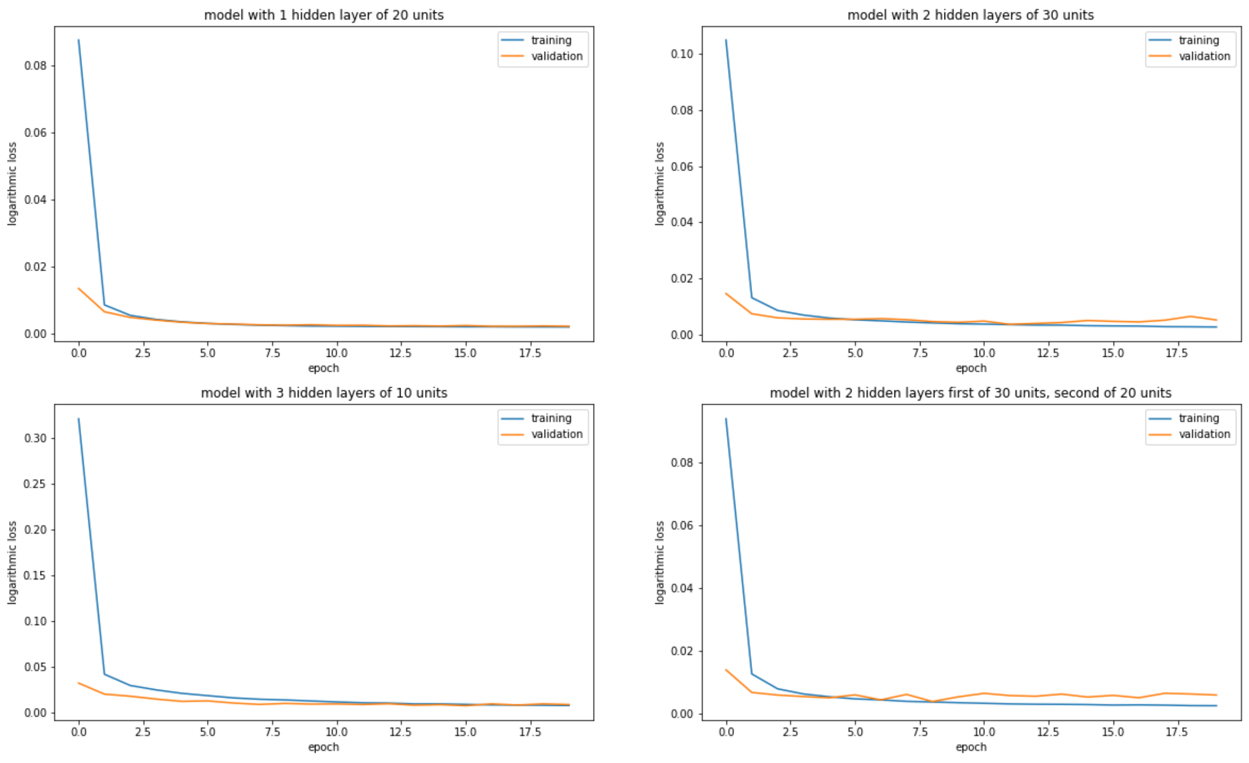


Figure 4: Plots of of some chosen model losses

d) Optimizer investigation

We run an exhaustive grid search in the optimizer and learning rate schedulers space by keeping fixed the best architecture from the previous set of experiments, which is a neural network with a single hidden layer of 20 units with ReLu activation. We test a vanilla SGD, SGD with momentum set at 0.9 and 0.75, and the Adam optimizer. For each of these, we vary the learning rate according to a variety of schedulers. The proposed learning rate schedules, visualized on figure 5, are:

- fixed 0.1,
- fixed 0.01,
- fixed 0.001,
- fixed 0.0005,
- exponential decay starting at 0.01 with decay rate 0.9,
- cosine decay starting at 0.01 to 0.001,
- piece-wise constant decay from 0.1 to 0.0005.

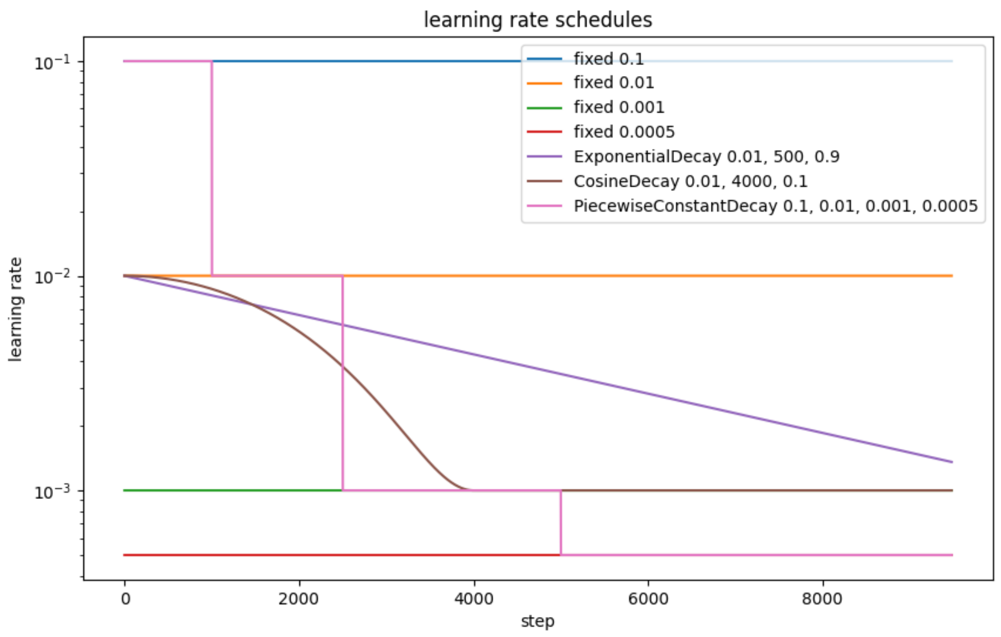
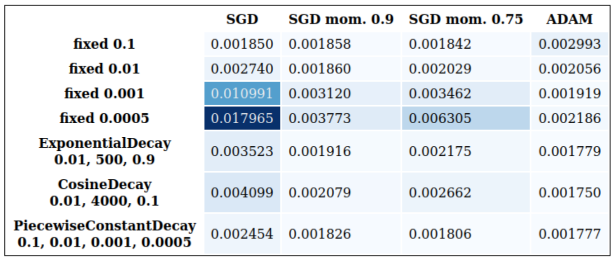


Figure 5: Loss on the validation data

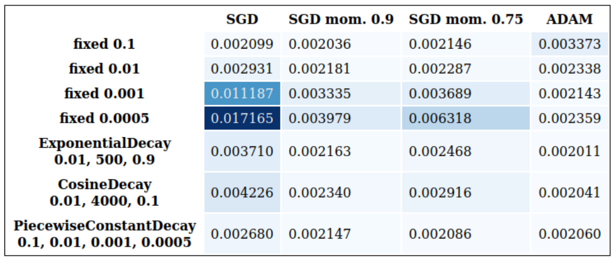
In the tables in figure 6 we report training and validation losses for all the combinations of optimizers and schedulers. We can see that most of them perform in a similar way, but we can also observe higher losses, when the learning rate is 0.001 or 0.0005, probably because it is too small and the models do not converge fast enough. Only the Adam optimizer seems to be able to mitigate the issue and handle also small learning rates effectively.

Last but not least, we observe that, although we are able to perform better than the benchmark linear model, the gap is not at all huge.



	SGD	SGD mom. 0.9	SGD mom. 0.75	ADAM
fixed 0.1	0.001850	0.001858	0.001842	0.002993
fixed 0.01	0.002740	0.001860	0.002029	0.002056
fixed 0.001	0.010991	0.003120	0.003462	0.001919
fixed 0.0005	0.017965	0.003773	0.006305	0.002186
ExponentialDecay 0.01, 500, 0.9	0.003523	0.001916	0.002175	0.001779
CosineDecay 0.01, 4000, 0.1	0.004099	0.002079	0.002662	0.001750
PiecewiseConstantDecay 0.1, 0.01, 0.001, 0.0005	0.002454	0.001826	0.001806	0.001777

(a) Learning rates schedules



	SGD	SGD mom. 0.9	SGD mom. 0.75	ADAM
fixed 0.1	0.002099	0.002036	0.002146	0.003373
fixed 0.01	0.002931	0.002181	0.002287	0.002338
fixed 0.001	0.011187	0.003335	0.003689	0.002143
fixed 0.0005	0.017165	0.003979	0.006318	0.002359
ExponentialDecay 0.01, 500, 0.9	0.003710	0.002163	0.002468	0.002011
CosineDecay 0.01, 4000, 0.1	0.004226	0.002340	0.002916	0.002041
PiecewiseConstantDecay 0.1, 0.01, 0.001, 0.0005	0.002680	0.002147	0.002086	0.002060

(b) Loss on the validation data

Figure 6: Losses after 20 epochs

e) Final model summary

For the final model, we stick with a neural network with a single hidden layer of 20 units with ReLu activation and use the Adam optimizer with exponential decay as a scheduler, which proved to perform the best in the previous experiments. We train with the full data set for 20 epochs and observe an error of 0.0018 on the training set at the end. The evolution of the training error during training is shown in figure 7. The learning curve looks very similar to the previous ones.

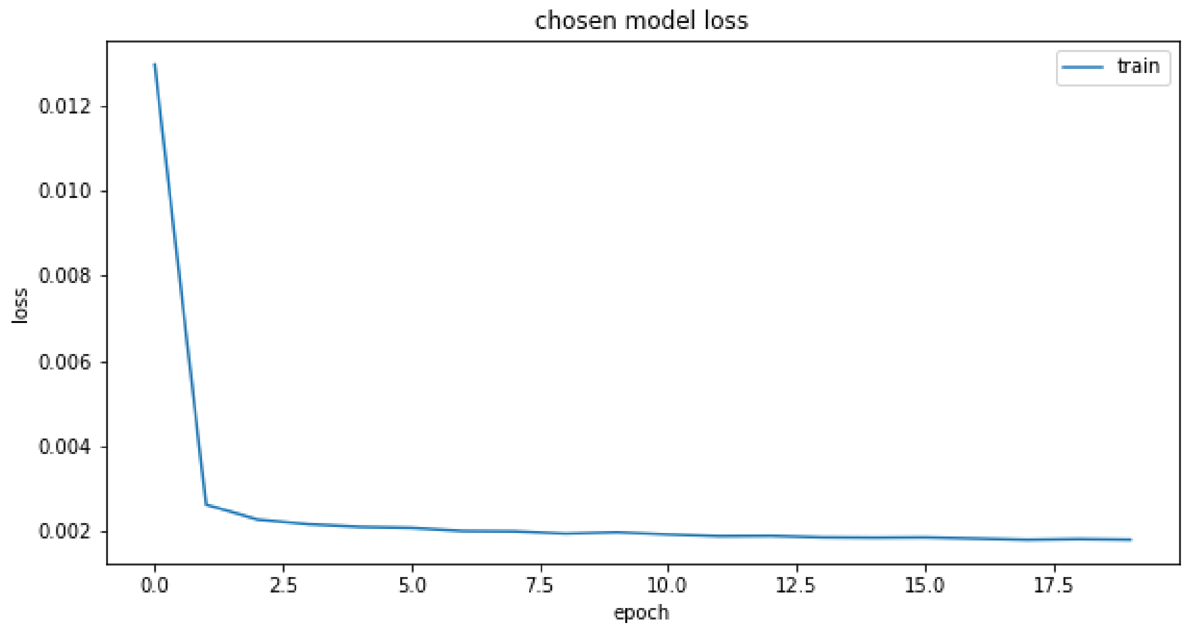


Figure 7: Loss on the training data

Predicting the test set also results in an error around 0.0019, which means the model performs quite well on the new, unseen data and it is not overfitting the train set. In the figure 8, the scatter plot of the true values vs predictions can be seen for the test data. The points are well placed around the identity line, and the errors are approximately Gaussian around zero as shown in the plot of the residuals.

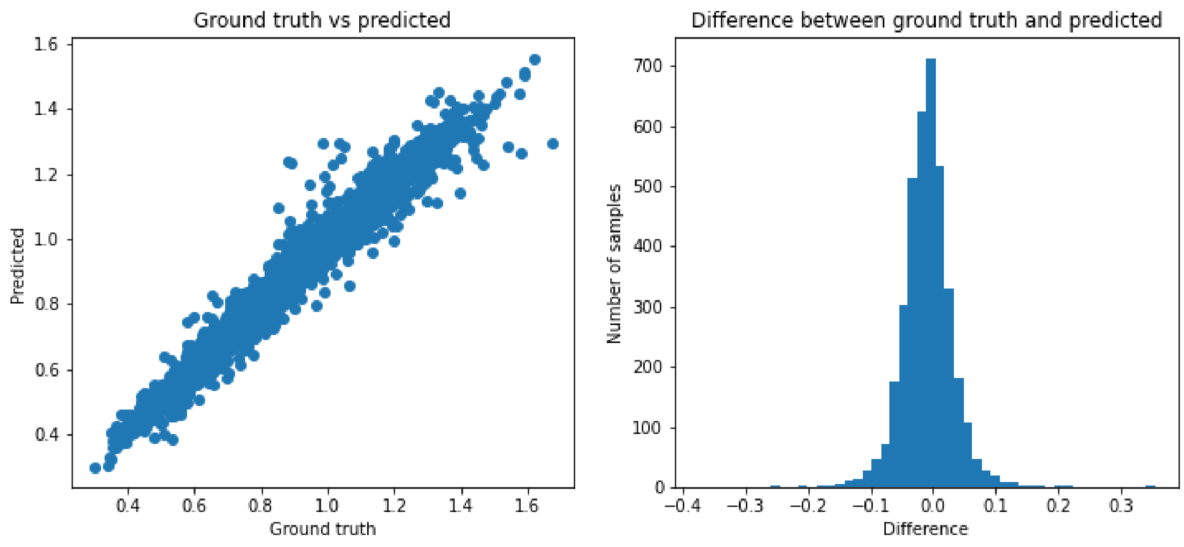


Figure 8: Final evaluation

References

[1] Chetty, R., Jackson, M. O., Kuchler, T., Hendren, J. S. N., Fluegge, R. et al. *Codebook for Publicly Available Data on Social Capital*. July 2022 [cit. 2022-11-12]. Available at: https://s3.us-east-1.amazonaws.com/hdx-production-filestore/resources/fbe5b0b9-e81c-41c7-a9f2-3ebf8212cf64/data_release_readme_31_07_2022_nomatrix.pdf?AWSAccessKeyId=AKIAXYC32WNARK756OUG&Signature=Pd14FQlgEG0ZLcjdzunNDhi7VqE%3D&Expires=1668250009.