EE 519: Deep Learning Theory & Fundamentals **Spring 2023**

Mini Project 1

Due: April 28, 2023, 11:59PM PT

Student Name: Sandy Dash *Instructor Name: John Lipor*

Problem Description

The goal of this project is to develop a predictive model based on Artificial Neural Network (ANN) which considers various geological features of different locations and accurately predicts geothermal favorability also known as heat flow residual for those locations. Accurate prediction of geothermal energy favorability allows for producing electricity by using earth's heat.

We are given a training dataset of 3112 samples which has 28 input features to produce one output that is heat residual. Because ANNs are considered as universal function approximator, I will use a Multi Layered Perceptron (MLP) model for this project to examine the efficacy of ANNs for this dataset.

Exploratory Data Analysis (EDA)

Data defines the architecture and the algorithms required for a machine learning project. But because I don't have a background in Statistics, I was unaware of what problems in the data can lead to poor performance of the model. For example I know that outliers are bad for the model but if dataset is noisy while the model is trained on a very clean dataset then the model will not be a good predictor. Having an intimate understanding of the features and the field also helps alot with EDA because then one will have an understanding about what features and trends in the data are helpful for a certain prediction.

To familiarize myself with the field, geological features and research direction in this field I read Dr. Lipor's previous work in this field [1].

Because of the above mentioned gaps in my knowledge I started this project with a vanilla ANN and considered Root Mean Square (RMSE) for validation set as a metric of performance. RMSE is calculated based on raw values of heat residual prediction as opposed to binning and ranking them as described in project description. I have considered this project as a pure regression problem. Some of my initial steps are described below:

Initial approach to understand type of dataset

1. Paper [1] concluded that out of the five feature categories they found two types of feature categories that are dominant predictor of heat residual, those are 1) Heat Flow 2) Distance to the fault. So I build a vanilla ANN considering 10 features out of the 28 given features that match the above two categories. For this model, the validation data is not even hidden from the model so I expect the model to perform very well. **RMSE 524.7**
2. For the above model even if I consider all 28 features **RMSE 526.4** which means at this stage considering reduced features was not boosting the performance. Therefore I chose to consider all 28 features.
3. I was normalizing the inputs and labels for this model based on the following equation.

$$norm\_input = (x - mean)/std \quad (1)$$

where x is sample, mean and standard deviation (std) area taken for entire training dataset.

4. I used the describe method of pandas dataframe and looked at the spread of the features. The features have a diverse spread for e.g. measure of central tendency for all the features has a range [3:13,102] and range of std is [0:44,346]. Therefore I decided to normalize the inputs and labels considering feature-wise mean and std. **RMSE 551.8**

5. Finally I studied about outlier removal techniques from [2,3,4] and chose Inter Quartile Range outlier removal technique considering the feature-wise mean and std. **RMSE 80.2**

By this point it was clear that I need to visualize this dataset and apply effective outlier removal techniques. I read through [5,6] and visualized all the features. Refer to figure 1 through figure 8 for the plots for this section.

Justification for outlier removal technique and data transformation

For outlier removal I experimented with two techniques; 1) Zscore 2) Inter Quantile Range (IQR) both performed with feature-wise mean and std. For ANN to generalize better, it must not see the validation dataset, therefore I split the training dataset into train and val sets with the following sizes.

Original	3112x29
Train	2490x29
Val	622x29

Technique	Training Data Size
Z score	1813x29
IQR	856x29

IQR technique has a 65.6% of data reduction whereas Z score method has 27% data reduction. I chose the z score method for outlier removal because it provides more training examples to the model.

I decided to remove the data normalization based on equation 1 and perform data transformation. I tried log transformation and quantile transformations. Log transformation was not changing the feature distribution to normal as quantile transformation with 'normal' argument. Normal distribution of features was also improving performance of the model. Quantile transformation was scaling the dataset between 0 and 1.

Challenges

The biggest challenge for this project is the time constraint. We were only given one week to understand the problem statement, do some background research, understand the dataset and build a model that can have the best performance on this dataset. Having said that, other specific challenges of this project are as described below:

1. **Not having any background in geology or geological data or research in this field.** This meant I had to spend some time understanding what these features mean and familiarize myself with current research in this field.

2. **Not having any Statistical background.** Instead of building a vanilla ANN and trying various experiements to figure what out the challenges with the dataset, I should have done EDA first but I was unaware of how badly the performance could be affected by outliers and skew in distribution of the dataset. So I learned this the hard way.

3. **No benchmark to beat.** It would had been better if we were given a benchmark to beat. I think that could help with navigating through the plethora of choices in EDA techniques and ANN architectures.

Since I am using my own laptop with Ubuntu 22.04 OS and an anaconda environment which is already setup, thankfully I didn't have any issues with setup. Also after applying early_stop algorithm the model begin to converge in less than 1000 epochs therefore I didn't even need to use CUDA. All the processing was done using Intel Iris graphics card.

Approach

Since I have already justified my EDA choices up above, in this section I will describe and justify the choices for my model architecture. This section is described using two types of meteric for the model performance.

Performance meteric for the model

1. **Accuracy**

$$\begin{aligned} Correct\_pred &= binY\_hat - binY \\ Accuracy &= (Correct/Validation\_size) * 100 \end{aligned} \tag{2}$$

   where binY and binY_hat are labels and predictions respectively, which are ranked and binned as described in the project document.Accuracy is measured in percentage.

2. **Mean Ordinal L1 Loss**

$$\ell(y, \hat{y}) = (|y - \hat{y}|)/Validation\_size, \tag{3}$$

   where $y, \hat{y} \in \{1, \cdots, 4\}$.

Model architecture and justification

1. **Data Split** In order to see if the model is generalizing, training data must be split into train and val sets. I have chosen a 80:20 % split between train and val sets which allows for enough training and validation samples despite the 27% training size reduction due to outlier removal. Validation data is completely hidden from the model.

2. **EDA on train set only** As indicated above in the EDA section that without any outlier removal, the model was performing very poorly. I also noticed that I get about 10% performance boost for a normal distribution of dataset as opposed to skewed dataset. But I wanted the model to generalize better therefore I chose to remove outliers and change the distribution of training set while only changing the distribution of validation set using quantile transformation and **did not remove outliers**.

3. **Input and output size** Input size is 28 and output size is 1.

4. **Number of hidden layers** Paper [1] chooses three hidden layers for a dataset of over 700,000 examples therefore I have chosen three hidden layers. I have also tested with four and five hidden layers but there was no significant performance boost.

5. **Weight and bias initialization** I chose Xavier init for weight initialization with ReLU activation function as argument and random normal distribution between 0 to 1 for bias initialization. These initialization provides about 5% performance boost to the model.

6. **Activation function** I have chosen ReLU activation function because its gradient is a step function. Positive values are favored over negative values which aligns with what instructed for the project that we are more interested to find High and Very High heat residuals compared to Low heat residuals. All values below 25 are binned into low heat residuals.

7. **Loss function** Initially I chose MSE Loss function but I got the best performance with Huber loss i.e. SmoothL1Loss function of PyTorch library. This is because Huber loss is less sensitive to outliers in the data compared to squared loss.

8. **Optimizer** In PyTorch model the optimizer function updates the gradents to minimize the loss. For my research project which is also a regression problem, I had tested all of the optimizers. Adams algorithm performs better for regression problem therefore I have chosen Adams algorithm.

9. **Overtraining prevention** I have chosen early_stop method to prevent over training which has a parameter called patience which means model is tolerant to how many instances of increase in validation loss. Training and validation sets are examined in every epoch loop. Once patience expires, the model breaks out of the epoch loop.

10. **Data Loader** Initially I was training all of the training data in one epoch but this was causing an inconsistencies in performance metric. The model execution time was very fast but I could not trust the model performance because of the inconsistency. When I added training dataloader with a configurable batch size while validation dataloader goes through the val size examples in each epoch, I saw about 8% performance boost and consistency in performance.

11. **Hyperparameter tuning** I have used Optuna for hyper parameter tuning for 100 trials.

12. **Hyperparameters and their values for best mean ordinal L1 loss** Hyperparameters in this project are:
    Number of epochs - 4795
    Number of neurons in hidden layer as known as hidden size - 20
    Patience - 319
    Learning Rate - 0.008335259958643284
    Batch Size - 15
    Model converges in less than 1000 epochs.

Evaluation and summary

Many of the questions asked in this section have been described above because I thought the report flows better in that way. Here I will describe the results and present figure for training and validation loss. One thing that I didn't have time to do was; determining feature correlations and trying with different number of input features based on correlation heat map. As stated above I had tried with most dominant features that the paper [1] had concluded, but that was hurting model performance so I chose to consider all 28 features. Refer to figure 9 and figure 10 for the plot of training vs. validation loss.

With the above mentioned tuned hyperparameters I got the best performance on validation set as described below:

Accuracy - 57%
Mean Ordinal L1 loss - 0.622

One thing I noticed is that if I change anything in model architecture e.g. loss function or optimizer algorithm then the model needs to a re-tuning of hyperparameters.

What I learned

Here is a summary of what I learned from this project.

1. **What does an accuracy of 57% mean?** Despite all of the EDA and model architecture choices the **model can only predict correctly for just over half of the validation set**. Removing outliers in validation set will improve the performance even more but then we make an assumption that the test dataset should also be free of outliers. Binning the numerical predictions to ordinal ranks definitely helps in boosting the performance because we are no longer comparing numbers but bins instead.

2. **Can a modern neural network architecture provide any better model performance?** As stated above I have chosen MLP as the model architecture which is quite dated. It would be interesting to see if adding more convolutions can extract more meaningful information out of a noisy dataset.

3. **Tools I learned from this project -** I have successfully implemented batched learning using PyTorch **dataloader** while still being able plot training and validation loss. This was an issue for my research project so I had avoided the use of dataloader but now I realized that without batched learning the performance of a model is inconsistent which is something I had noticed for my research project. In the past I tuned hyper parameters linearly which is a time consuming and inefficient method. I learned that **Optuna** is very easy to add and it does all the hyperparameter tuning seamlessly. I observed that hyperparams tuned by Optuna also helps in stabilizing model performance.

Code

I wanted to give a professional look to my code so I have put file headers which explain purpose of the files and function headers which elaborates on the behavior of the functions. Within the functions the code is well commented. My code is fully parameterized and I feel proud of a clean and well documented the code. I have uploaded this project on my github page : https://github.com/SandyDash19/GeothermalEnergyPrediction

References

This section is not written in IEEE style of citations because I am new to LaTex.

1. S. Mordensky, J. Lipor, J. DeAngelo, E. Burns, and C. Lindsey, "When less is more: How increasing the complexity of machine learning strategies for geothermal energy assessments may not lead toward better estimates," Geothermics, 2023.

2. EDA reference: https://www.analyticsvidhya.com/blog/2021/02/introduction-to-exploratory-data-analysis-eda/

3. Outlier removal reference: https://www.analyticsvidhya.com/blog/2021/07/how-to-treat-outliers-in-a-data-set/

4. Inter Quartile Range (IQR) method: https://www.analyticsvidhya.com/blog/2022/09/dealing-with-outliers-using-the-iqr-method/

5. Data visualization tutorial: https://www.kaggle.com/code/imsanjoykb/data-visualization-tutorial-matplotlib-seaborn

6. Box plot tutorial: https://proclusacademy.com/blog/quicktip/boxplot-separate-yaxis/

Figures

In this I have described everything in figure captions. I have also referred to figure numbers in the sections above. The document looks cleaner with figures at the end and therefore I chose to add all the figures here. Ideally I would have liked to display before EDA and after EDA pictures side by side but that is too time consuming therefore I have put the pictures one after other.

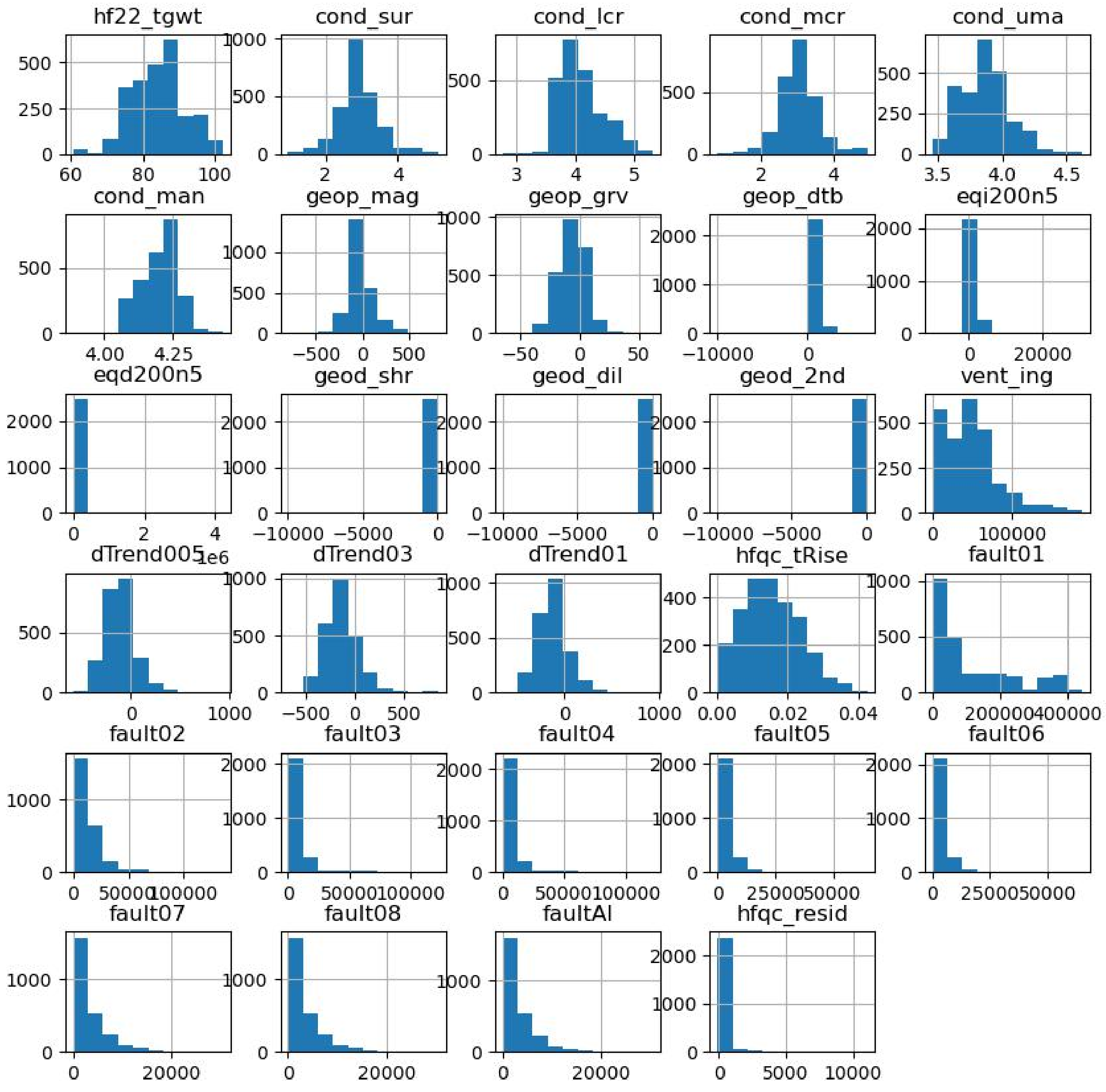

Figure 1: Distribution of all the features in training dataset **before** outlier removal and data transformation

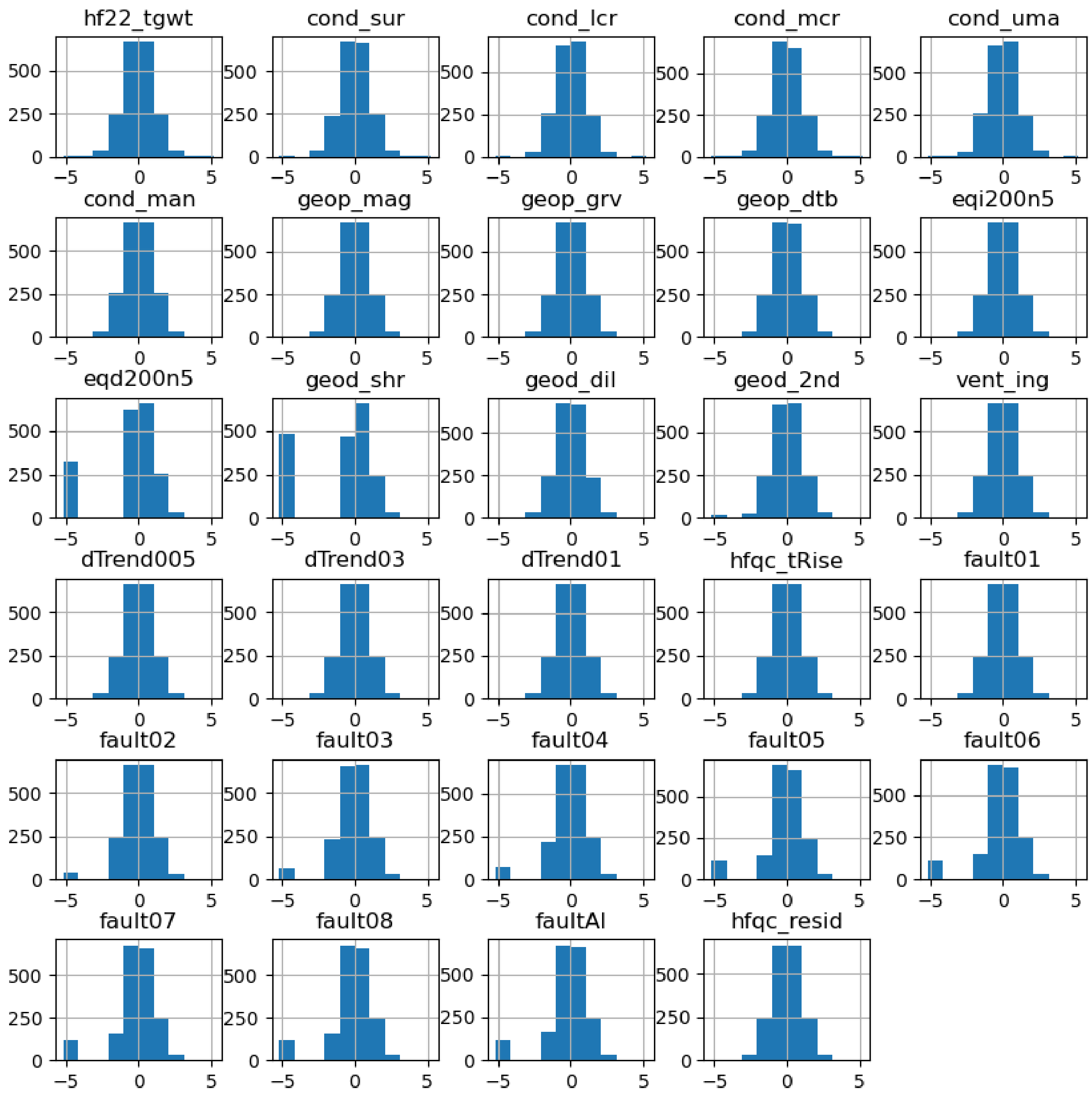

Figure 2: Distribution of all the features in training dataset **after** outlier removal and data transformation. We can see that features eqd200n5, geod_shr, fault02, fault03,fault04, fault05, fault06, fault07, fault08 and faultAl still has outliers and some of the data appears in the left side of the plots

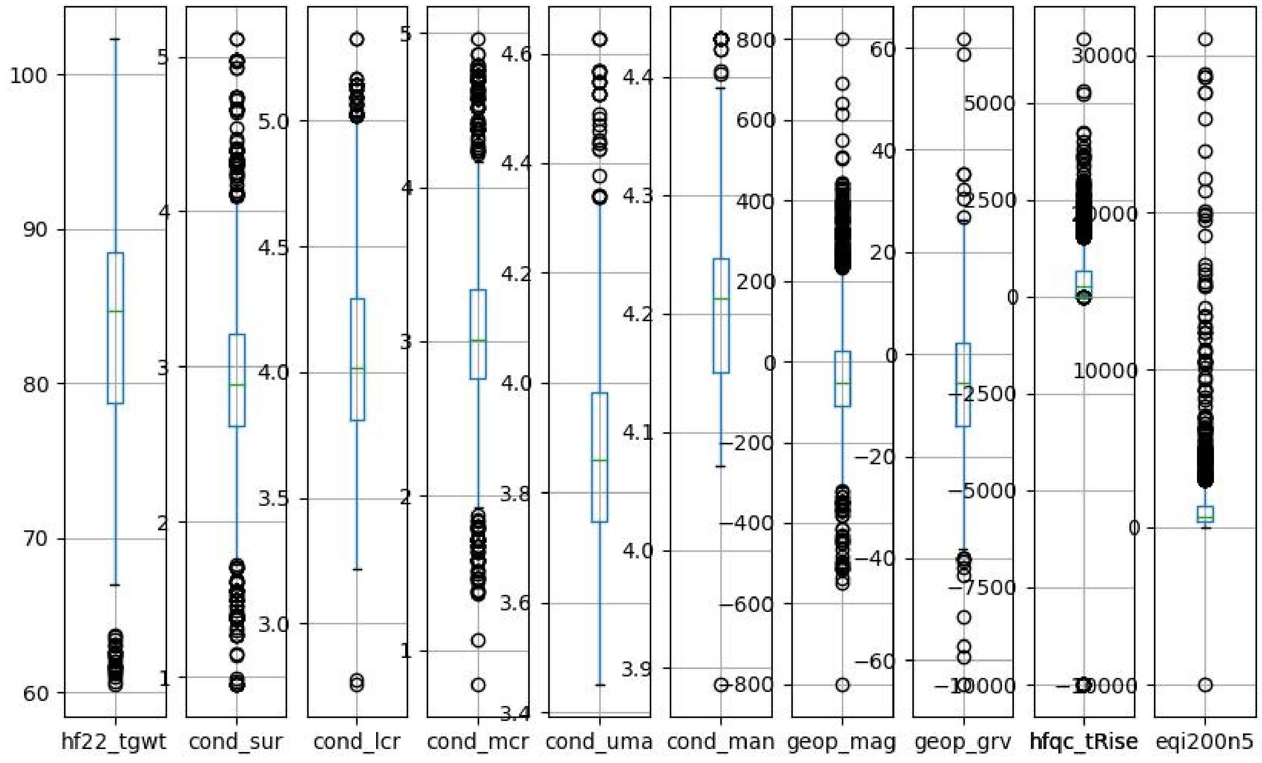

Figure 3: Box plot 1 **before** outlier removal and normalizing distribution. I have plotted the features in separate box plots for better clarity. We see that some features suffer from more outliers than the others and outliers also drastically skew the data set. As noted in the caption of figure 2, eqd200n5 has such skewed data set that most of the data is outside the mean and 25 and 75 percentile of the distribution

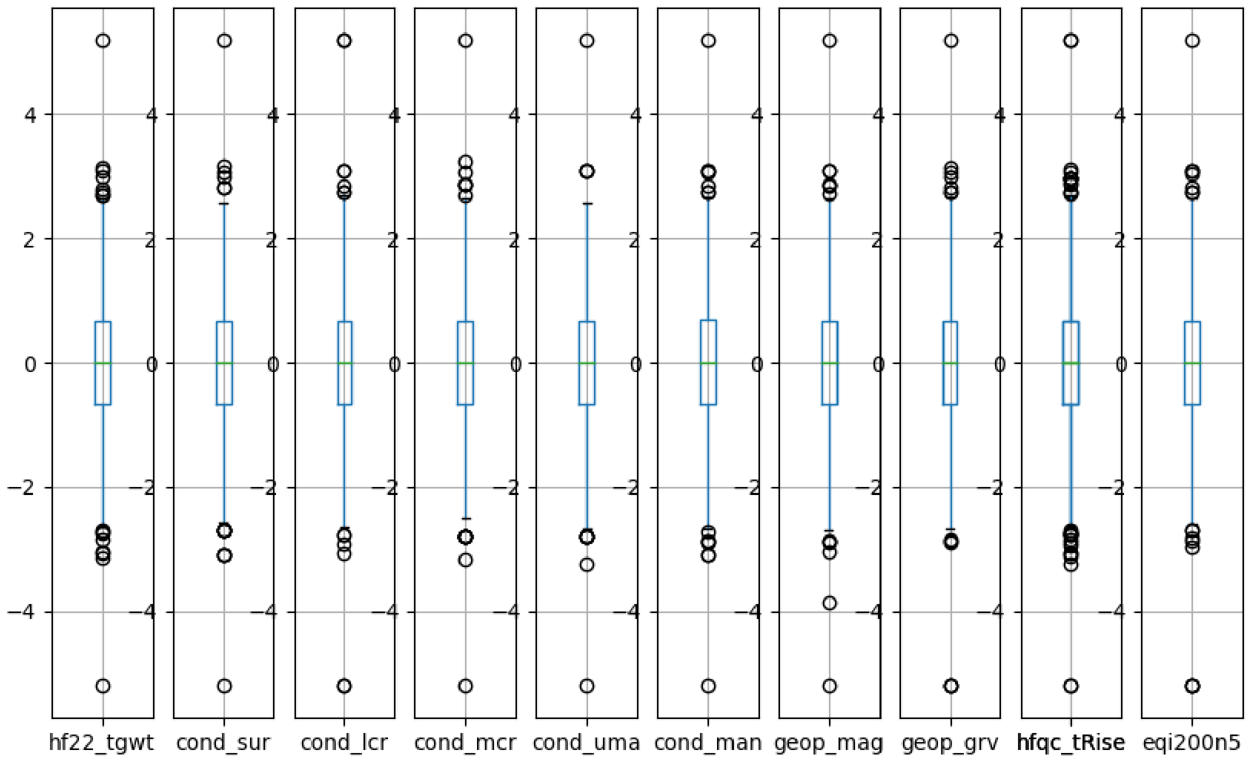

Figure 4: Box plot 1 **after** outlier removal and normalizing distribution

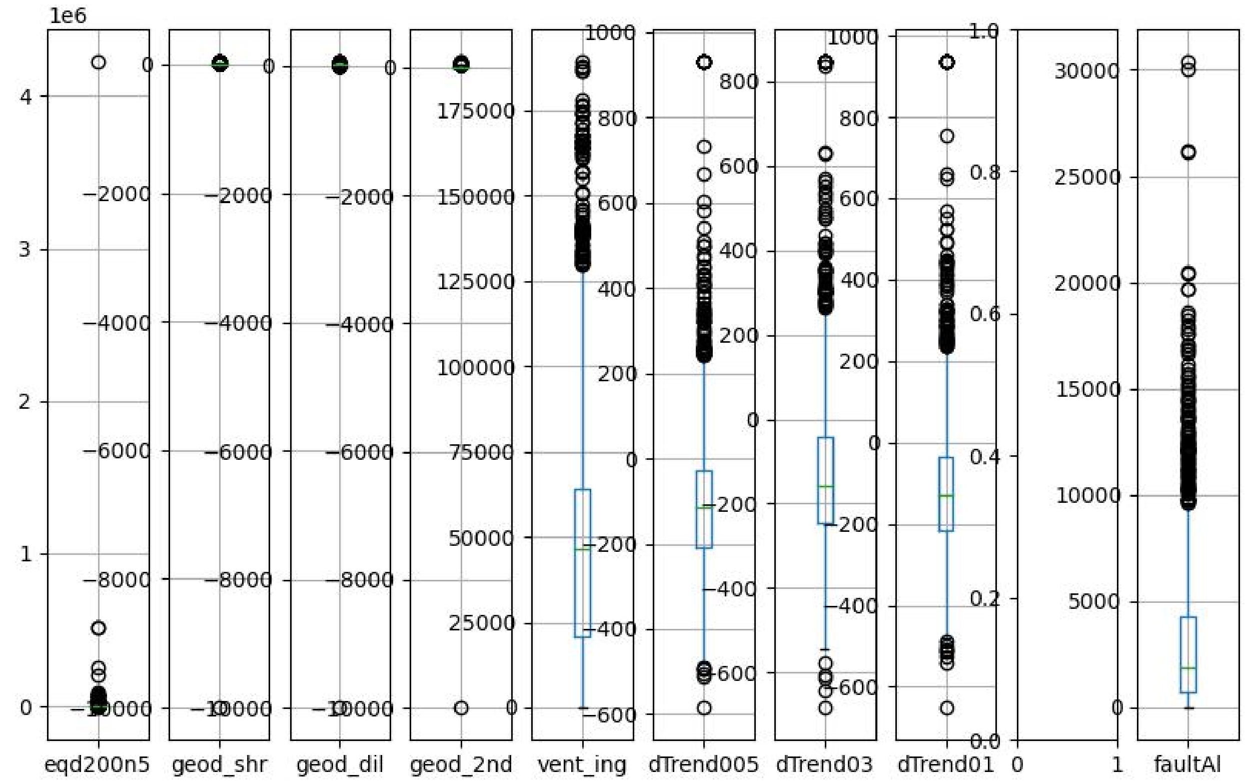

Figure 5: Box plot 2 **before** outlier removal and normalizing distribution

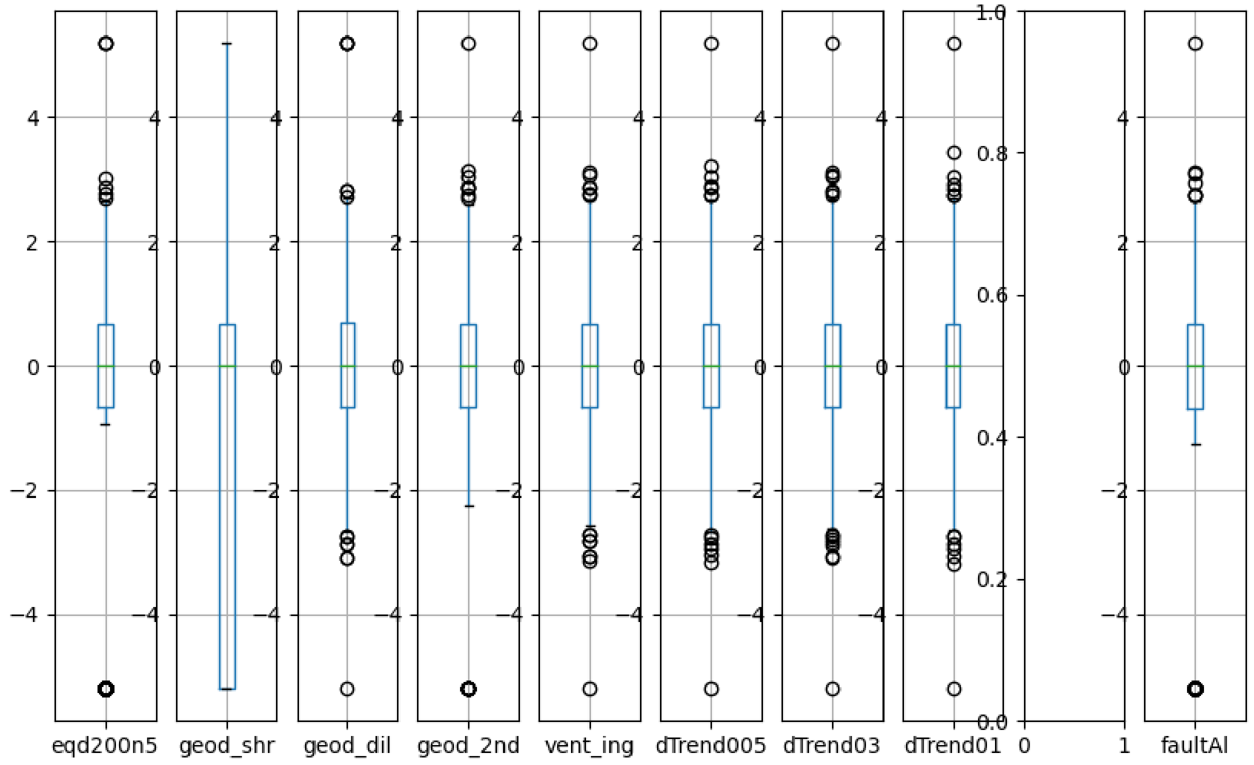

Figure 6: Box plot 2 **after** outlier removal and normalizing distribution

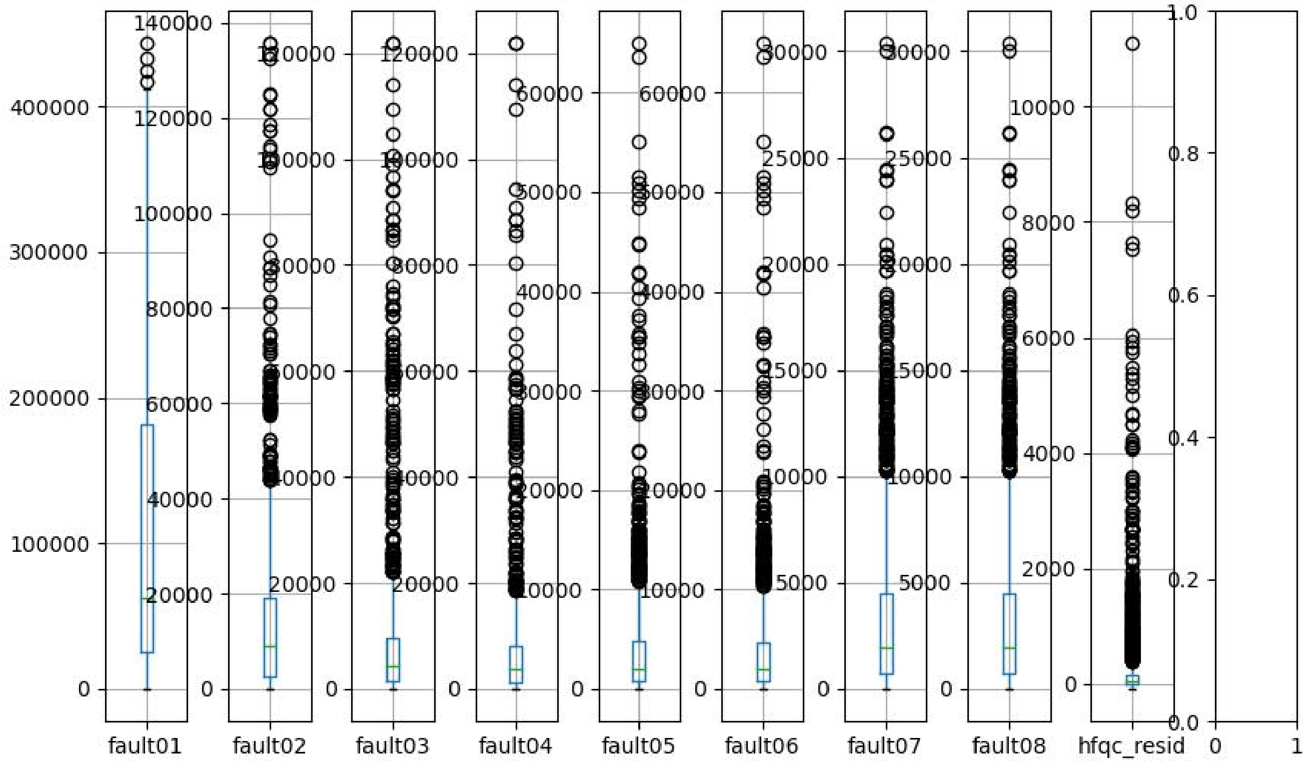

Figure 7: Box plot 3 **before** outlier removal and normalizing distribution. This category of feature i.e. distance to fault is especially noisy and skewed.

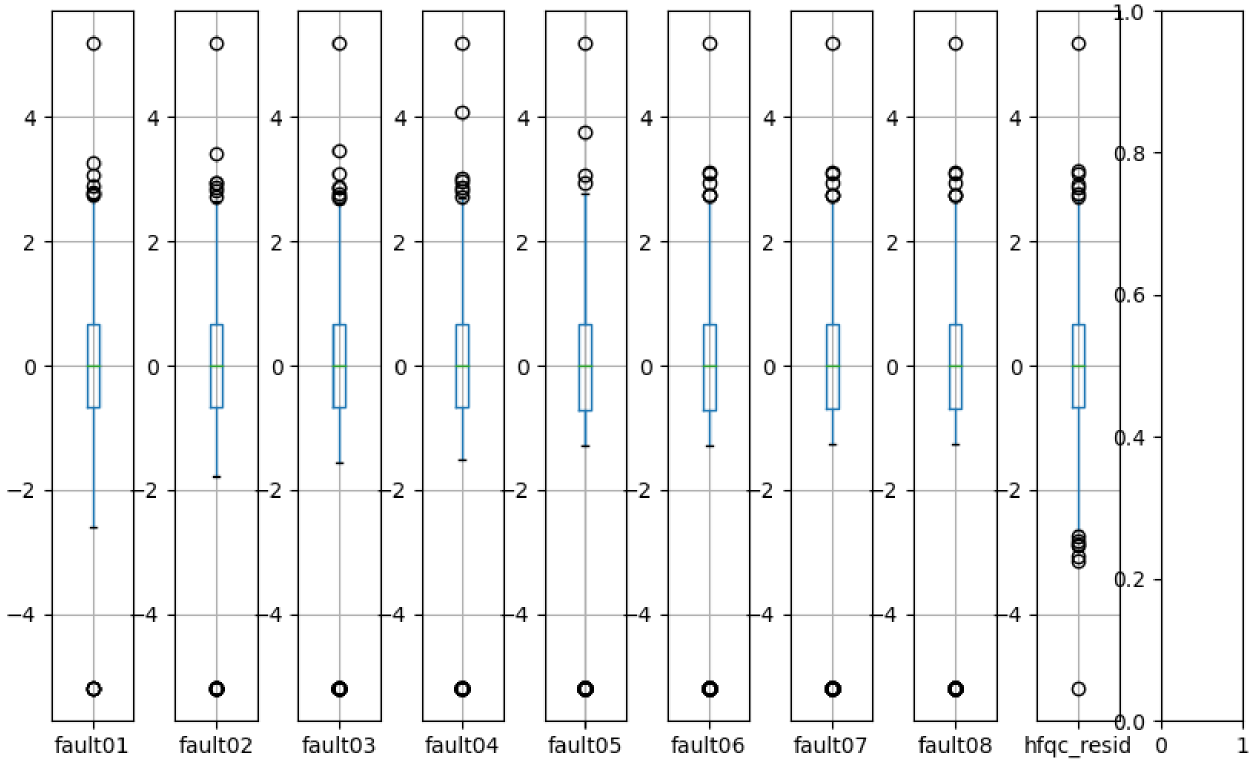

Figure 8: Box plot 3 **after** outlier removal and normalizing distribution

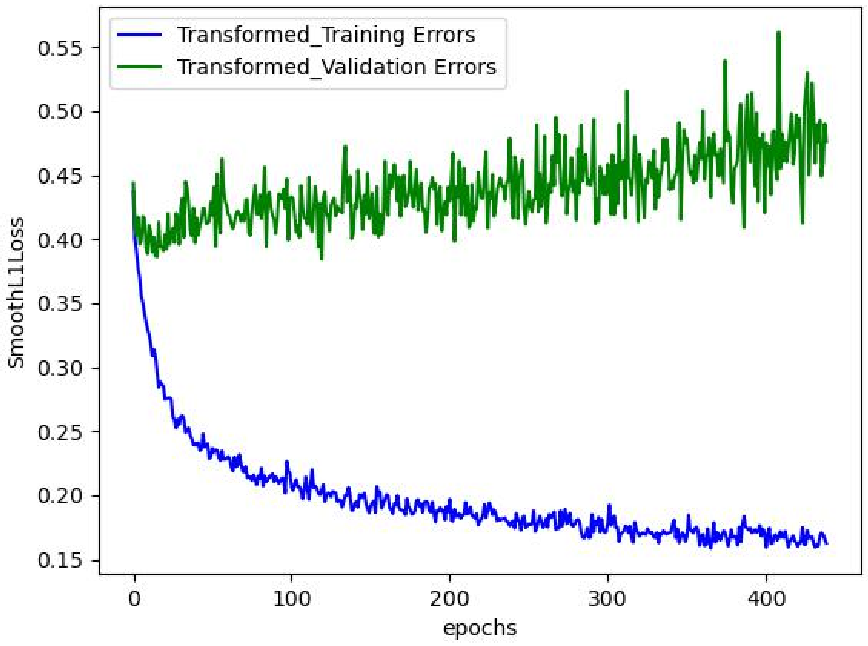

Figure 9: Training vs Validation Loss. Training and validation losses are averaged over their respective data loader sizes and appended into a list for every epoch. Since the train and val sets are transformed, the loss is also transformed. I could not find a way to smooth out the noise in the plot. May be averaging this loss over multiple runs could produce a less noisy image. We can see that the training data has the desired downward trend which means the model is learning and minimizing the Huber loss but the validation set does not show any such trend even though it should also show a similar trend. I wonder if its because the validation set did not go through outlier removal. If I don't do batched learning I get smooth plots with downward trends for both training and validation losses as shown in figure 10.

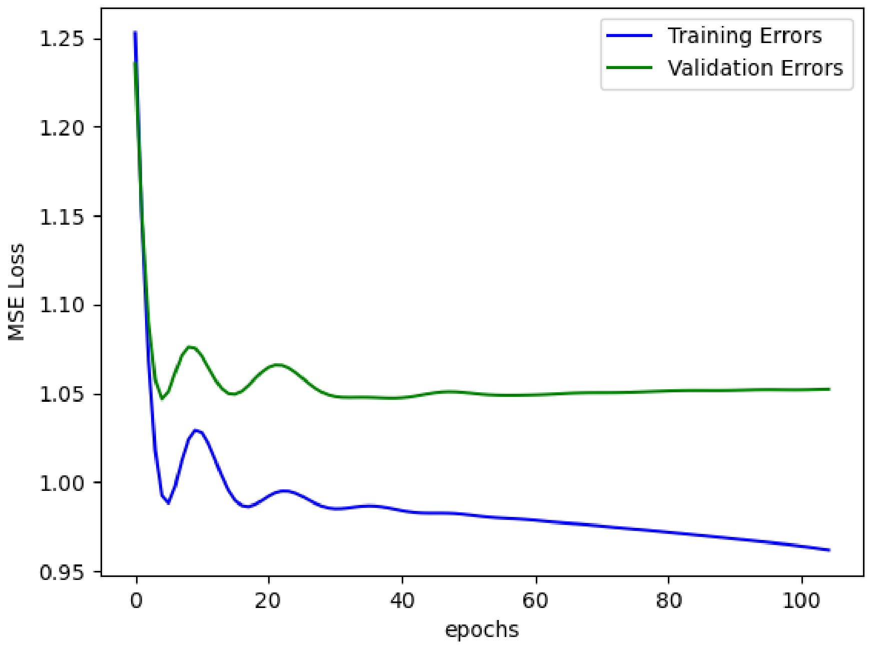

Figure 10: Training vs Validation Loss. This plot is captured while the model I was still tweaking the model but without batched learning. The purpose of this figure is to show that without batched learning, I get smooth plots for training and validation loss.