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

Mini Project 1

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

Student Name: Andy Franck *Instructor Name: John Lipor*

Problem Description

The objective of this project is to develop a neural network model capable of accurately categorizing geothermal heat flow residuals into one of four distinct classes. These residuals will be compared against actual heat flow measurements to assess the suitability of a geothermal energy site at the given location.

To accomplish this, a dataset consisting of 28 features, including heat flow residuals, will be used to train a neural network model. The model will be trained on a subset of the data, and then tested on the remaining data. The model will be evaluated based on its ability to correctly classify the heat flow residuals into one of the four classes.

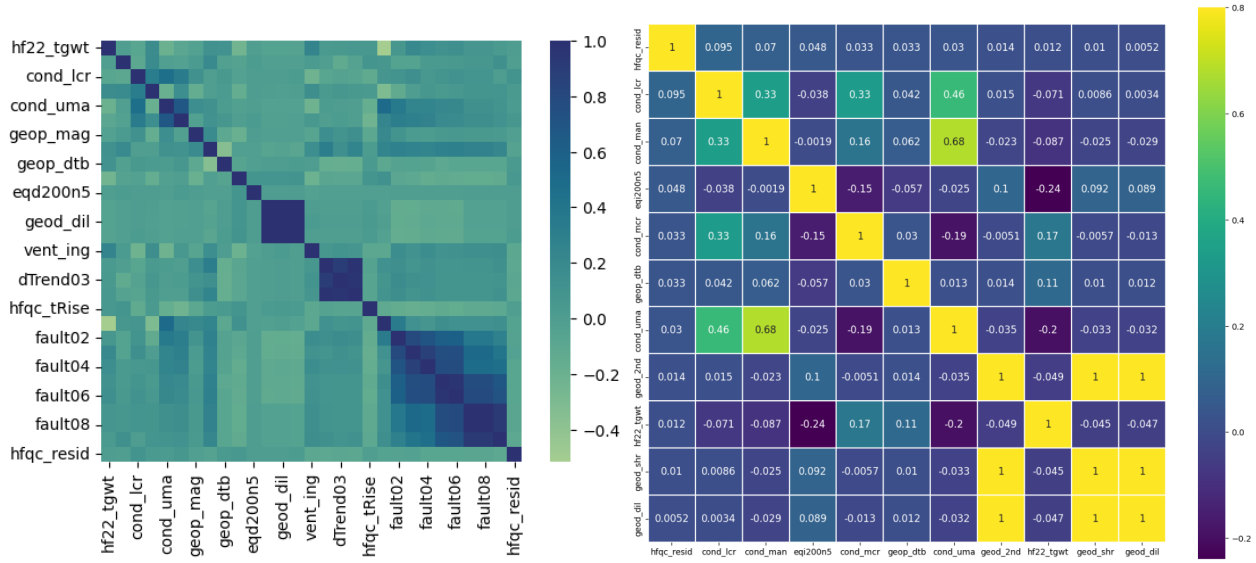
The training data is provided from both the United States Geological Survey [1], and the INGENIOUS project [2]. The data includes feature vectors provided by multiple surveys and sources. The data has been modified into an 80/20 training/validation split, and the validation data is separated from the model until it testing.

Although it would be convenient to have an easy-to-train model that can make accurate predictions, it is not guaranteed that any such model will work properly. This report aims to develop a model that has the highest possible correct classification rate with the given feature vectors and training data. Certainly, as our population continues to grow, finding economical sources of energy becomes more and more prudent. Developing a model that can accurately predict locations where a surrogate for geothermal energy is favorable would be an excellent development in our search for more sustainable energy.

Exploratory Data Analysis

To establish a baseline for exploratory data analysis (EDA), two articles [3], [4] were consulted in addition to a video tutorial by Rob Mulla [5]. The articles provided a general overview of EDA, and the video tutorial provided a more in-depth look at the process.

To gain a preliminary understanding of the correlations of all features/labels, a correlation plot was generated using the Seaborn library. These visualizations helped to identify high correlations between certain features and the labels, as well as between pairs of features.



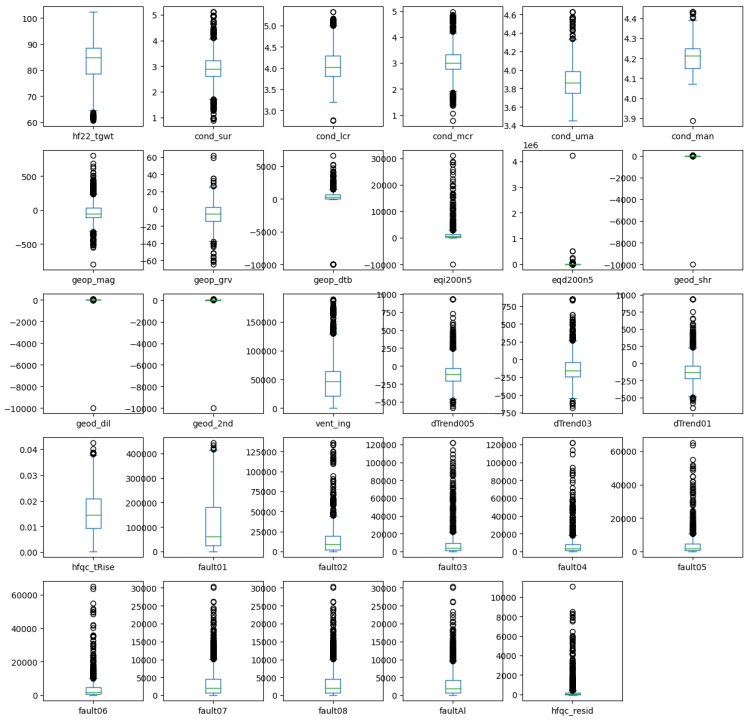
It is easy to determine from the images which features have a high correlation to each other. For example, *geop_dtb* and *cond_mcr* both have similar features, so it is not necessary to include both. Also notice *cond_lcr* has an extremely high correlation to the residuals, so it is included as one of the featuers.

Next, a pairplot was generated to visualize the distribution of each feature. This plot also helped to identify outliers and empty values. It also assisted in dermining how features related to each other.
PAIRPLOT HERE

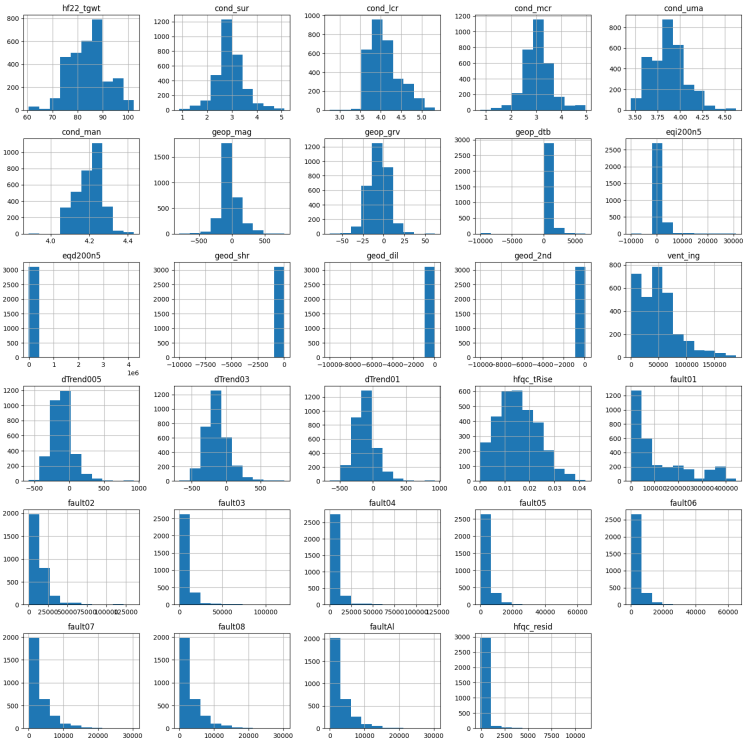
To get a more general description of the data, the pandas.describe() function was used. This function provides a summary of the data, including the mean, standard deviation, minimum, maximum, and quartiles.

	hf22_tgwt	cond_sur	cond_lcr	cond_mcr	cond_uma	cond_man	geop_mag	geop_grv	geop_dtb	eqi200n5
count	3112.000000	3112.000000	3112.000000	3112.000000	3112.000000	3112.000000	3112.000000	3112.000000	3112.000000	3112.000000
mean	84.215340	2.904011	4.096880	3.059997	3.867164	4.200543	-31.447629	-6.276163	442.590813	1271.432940
std	7.312246	0.561523	0.364093	0.532789	0.190964	0.065052	139.578571	11.761215	985.091432	2417.688062
min	60.543701	0.870828	2.758250	0.776779	3.450930	3.886300	-800.406006	-64.729401	-9999.000000	-9999.000000
25%	78.581848	2.617540	3.809910	2.764320	3.748020	4.150280	-108.173498	-14.096750	61.382524	332.500000
50%	84.735298	2.884190	4.018740	3.008610	3.859020	4.212105	-52.001501	-5.904085	279.886505	618.600006
75%	88.441725	3.210743	4.287540	3.335370	3.984090	4.247610	29.486575	1.835548	642.796753	1373.000030
max	102.328003	5.115060	5.325150	4.963910	4.626700	4.432190	800.979981	61.842201	6651.750000	31059.300780

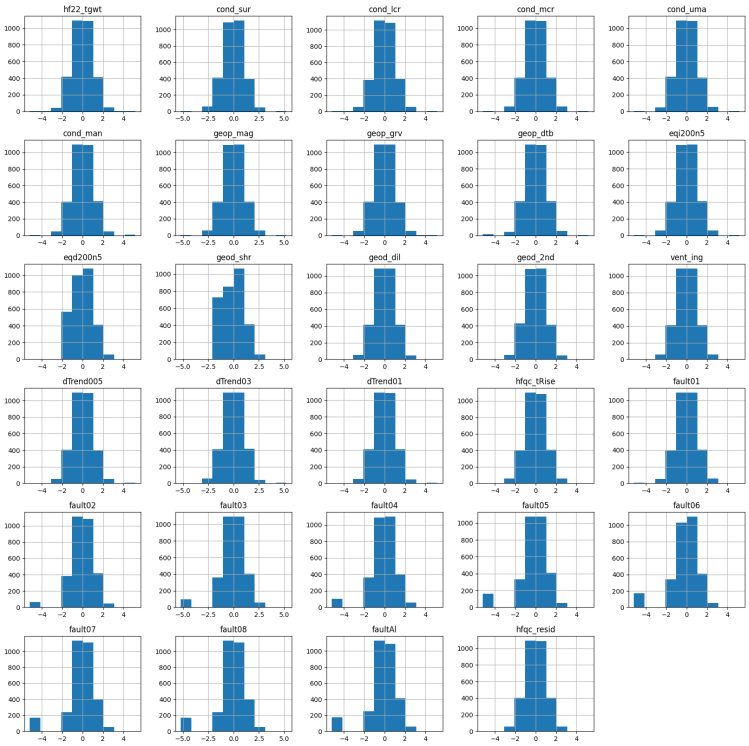
From this data, it is easy to determine which features contain outliers and empty values. For example, the features and *eqi200n5* have minimum values of -9999. These outliers need to be removed from the data to assist the model in training. To confirm visually, box and whisker plots were generated for each feature.



Finally, histograms were plotted for each feature to assess whether the data was normally distributed. Normality is crucial in the training of machine leraning models. If the data is not normally distributed, it can reduce the accuracy of the model dramatically.



To address the non-normality of the data displayed above, a quantile transform was applied to the features. This transform maps the data to a normal distribution.



Challenges

The data management phase presented the main challenges of the project. Despite the relative simplicity of the MLP itself, managing the data types and transformations proved to be the most persistent challenge.

One of the major obstacles was convering the data to types that the different libraries could accept. This required a lot of trial and error to identify the root of the data type issues. Moreover, managing transformations and ensuring they were properly performed and then reverted in the correct order was also a challenge.

The model's accuracy was often significantly impacted by my improper implementaions of the transformations and reversals. Additionally, performing the transformations in combination with the train/validation split was also a challenge. Often, some of the modifictations required a full dataset to be present, so it was necessary to save the original features, attach the output, perform the transformation, then pull out the output and save it separately. This would then be repeated for the validation set, and then reverted in the proper order.

Finally, the d2l library was also a source of significant frustration. Importing the library into Colab often resulted in obscure errors that were not the cause of incorrect code. As a result, the code was modified to use a pure PyTorch implementation instead of the d2l library.

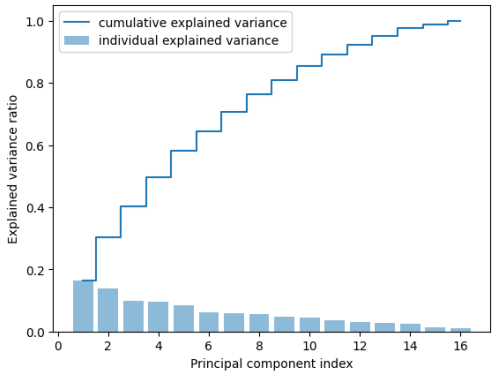
Approach

The initial step in the development involved creating a simple working model that analyzed a sample of the data. Although the results were not satisfactory, this model provided a baseline for the development of the

final model. After the baseline model was created, the model was then updated and trained on all 3112 samples and 16 features, which were chosen from the EDA applications and recommended features provided in the project description. The DataLoader function was used to load the data into the model.

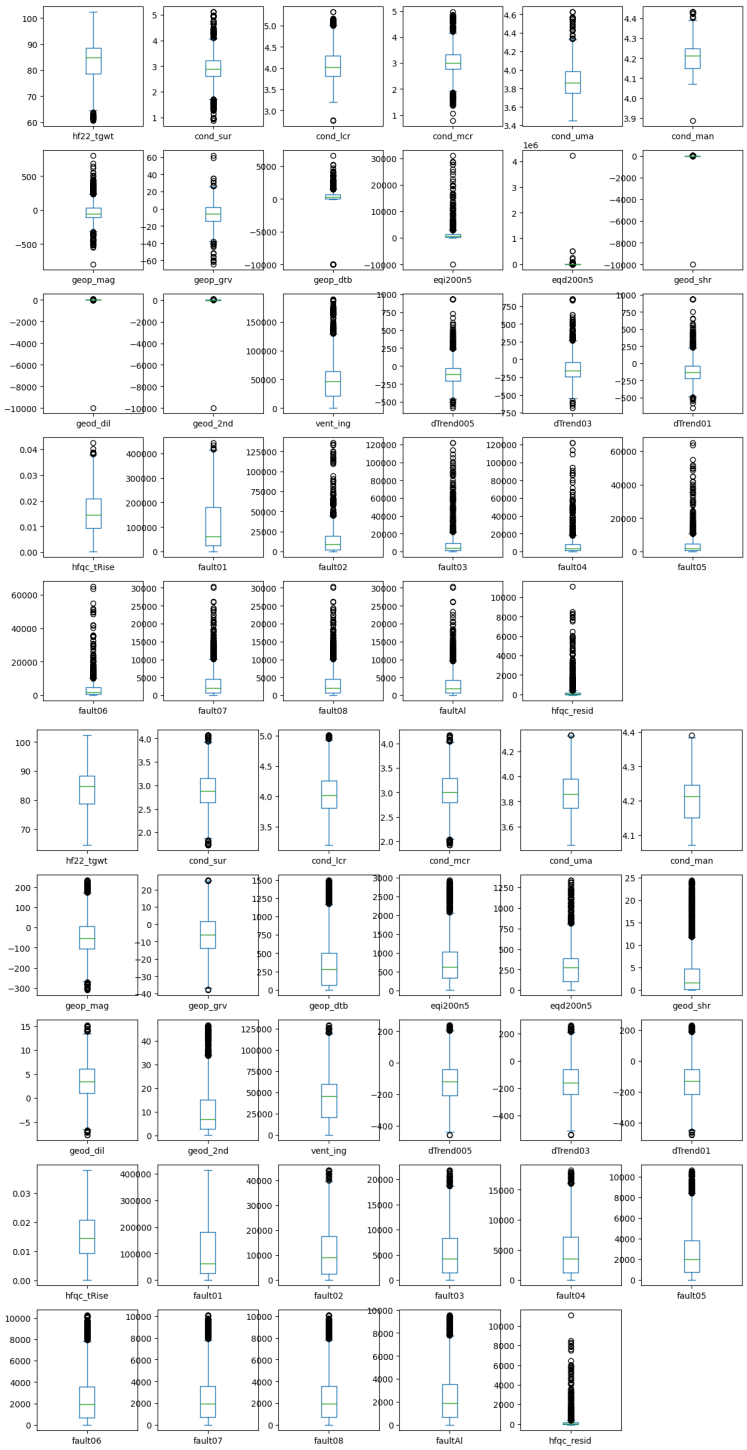
The preprocessing of the data did not require significant data wrangling, as the pandas DataFrame method used to load the data made it very easy to access and move. However, the data did require some transformations to ensure the model could properly train. The data was converted to a PyTorch tensor, and then the data was normalized using the mean and standard deviation of the training data. The application of each preprocessing method was fopllowed by a training and testing of the model for perforamce gain and accuracy, which was measure by Colab's built in timer aned the ratio of correct predictions to total predictions, respectively. By recommendation of [6], the testing/validation split was 80/20 percent, randomly shuffled. The validation set was kept hidden from the data during the training phase to avoid any bias, and all preprocessing methods were applied to it seperately.

To increase performance and reduce overfitting, PCA was applied to the training and validation sets to increase performance and reduce overfitting [7]. The first 8 principal components were found to account for almost all of the variance in the data, so the remaining components were dropped. However, saving all the principal components was essential to transform the data back to the original space for classifying the validation set.



Since the data was also not in normal distribution, a quantile transform was applied to the data to map it to a normal distribution (see EDA section). This was done to increase the accuracy of the model, as non-normal data can reduce the accuracy of the model significantly.

Since outliers were present in the data, they were removed using the IQR method [8]. A function was created to remove the outliers from the data. The function calculates the first and third quartiles (Q1 and Q3), as well as the IQR (Q3 - Q1), for each column. It then defines upper and lower thresholds for outliers as 1.5 times the IQR above Q3 and below Q1, respectively. Values outside of these thresholds are replaced with the median value of the column.



Because there were so many outliers, it was counterproductive to remove all of them. Instead, the outliers were replaced with the median value of the feature. This was done to avoid losing too much data.

Finally, the data was split into training and validation sets. The training set was used to train the model, and the validation set was used to test the model. The validation set was kept hidden from the model during the training phase to avoid any bias.

The model was then trained using the training set. It was trained for 10,000 epochs, at a batch size of 350. The hyperparameters were suggested based on the recommendations of [9]. These suggestions were inputted into Optuna [10], which is a hyperparameter optimization library. Suggestions for multiple different loss functions were intially provided, however Adam consistently outperformed the competition, so it was the clear choice, averaging nearly 10% higher accuracy averages than the second best alternative.

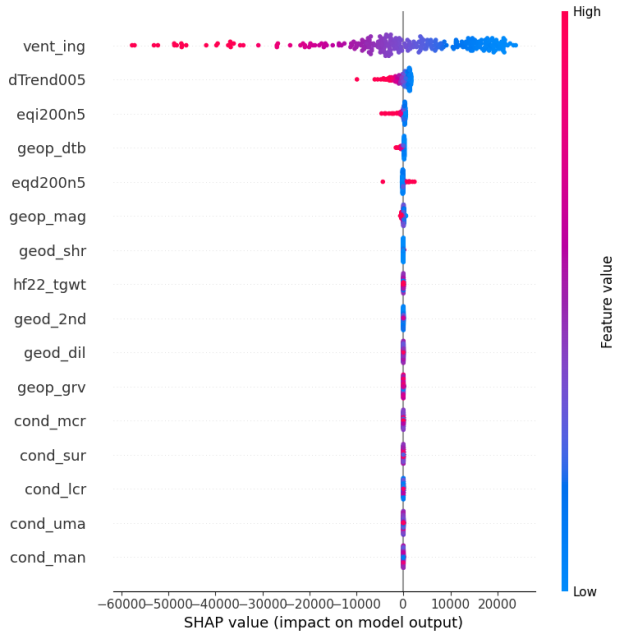
Evaluation and Summary

Unfortunately, the trained model did not perform to the level of accuracy desired. The model was able to achieve an average classification accuracy of 0.52%. Fortunately, although the model was not particularly accurate, it seemed to be consistent, as the accuracy was very similar for each of the 5 folds.

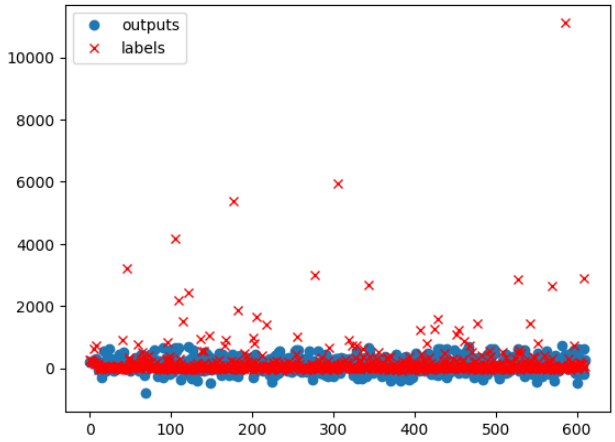
The model was also able to stay fairly consistent between both validation and test loss. Both values were very similar, although the validation loss was slightly higher. This is likely due to the fact that the validation set was not used to train the model, so it was not as familiar with the data.

```
Tester Loss after 10,000 epochs: 0.2204592
Validation Loss after 10,000 epochs: 0.2838920
```

The following SHAP graph is a good model of which features positively and negatively impaced classification. For example, the feature *vent_ing* either had a very positive or very negative impact on the classification. A feature like *dTrend005* appears relatively helpful for classification, while others appear not particularly useful at all.



The following graph shows the distribution of the predicted values. The model's predictions were fairly evenly distributed, with a slight bias towards the lower values. Because some label outliers were so high, the graph looks more skewed than it really should be after classifying.



Finally, the following is a confusion matrix of each classification class. The model seemed to classify the lower values fairly well, but struggled with the higher values. Perhaps limiting the higher values to a smaller cap would improve the accuracy of the model. Because some of the higher values were so high, they skewed the data and made it more difficult to classify.

Classes	True Positives	True Negatives	False Positives	False Negatives
Class 0	228	109	139	135
Class 1	531	4	61	15
Class 2	323	44	134	110
Class 3	347	50	70	144

What I Learned

This project provided an excellent opportunity to gain practical experience designing and training a nerual network on real data. The process of implementing most of the code from scratch and relying on documentation/other online resources allowed a deeper understanding of the process. The use of Optuna [10] was particularly valuable and provided a great introduction to hyperparameter optimization.

Additionally, this project served as as great practice for improving both EDA and general management skills, including data wrangling and data manipulation such as graphing and modifying data to gain insights on their behavior. Resources such as the PyTorch YouTube channel and online tutorials such as those referenced in the bibliography were invaluable in developing a strong understanding of the basics of neural networks and PyTorch.

References

[1] O. of Energy Efficiency and R. Energy, "Geothermal basics," 2022, https://www.energy.gov/eere/geothermal/geothermal-basics.

[2] G. B. C. for Geothermal Energy, "Ingenious project," 2022, https://gbcge.org/current-projects/ingenious/.

[3] "A simple tutorial on exploratory data analysis," 2022, https://www.kaggle.com/code/spscientist/a-simple-tutorial-on-exploratory-data-analysis.

[4] "A comprehensive guide to data exploration," 2016, https://www.analyticsvidhya.com/blog/2016/01/guide-data-exploration/.

[5] R. Mulla, "Exploratory data analysis with pandas python," https://youtu.be/xi0vhXFPegw.

[6] R. Pramoditha, "Why do we need a validation set in addition to training and test sets?" 2022, https://towardsdatascience.com/why-do-we-need-a-validation-set-in-addition-to-training-and-test-sets-5cf4a65550e0.

[7] L. Li, "Principal component analysis for dimensionality reduction," 2019, https://towardsdatascience.com/principal-component-analysis-for-dimensionality-reduction-115a3d157bad.

[8] N. Sharma, "Ways to detect and remove the outliers," 2018, https://towardsdatascience.com/ways-to-detect-and-remove-the-outliers-404d16608dba.

[9] L. Shukla, "Designing your neural networks," 2019, https://towardsdatascience.com/designing-your-neural-networks-a5e4617027ed.

[10] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.