ML Final

I. Introduction

In this competition, I first naively tried tree based models and do some dataset mining. Then, I compared various data preprocessing methods and tried Multi layer perceptron (Neural Network) to get optimal performance. At the end, I get 0.5908 in private test set using a 22-layered Neural Network with minimal data preprocessing (to preserve the originality).

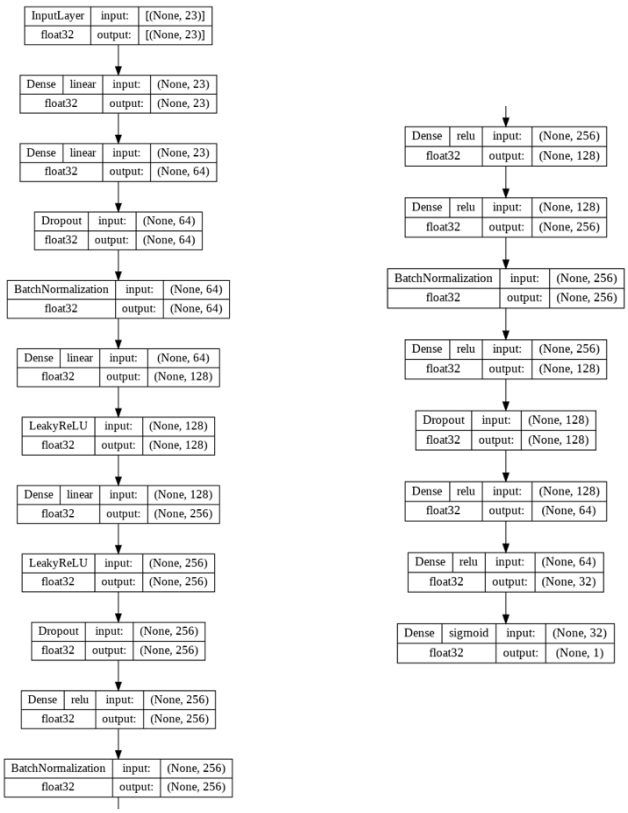
II. Environment Information (Python = 3.8.16)

1. Packages
   - A. tensorflow = 2.9.2
   - B. xgboost = 1.7.3
   - C. numpy = 1.21.6
   - D. catboost = 1.1.1
   - E. pandas = 1.3.5
   - F. sklearn = 1.0.2
2. Model Weight Download Link
   1. Multi Layer Perception (Download the whole folder)
      https://drive.google.com/drive/folders/10O3dMZA7Qb_t3gNSLPHH2Zoli_4jUWRi?usp=share_link
   2. Residual MLP (Download the whole folder)
      https://drive.google.com/drive/folders/1PMAyz8mLhnI7WfcMMm7LLzCagQ4qmDny?usp=share_link

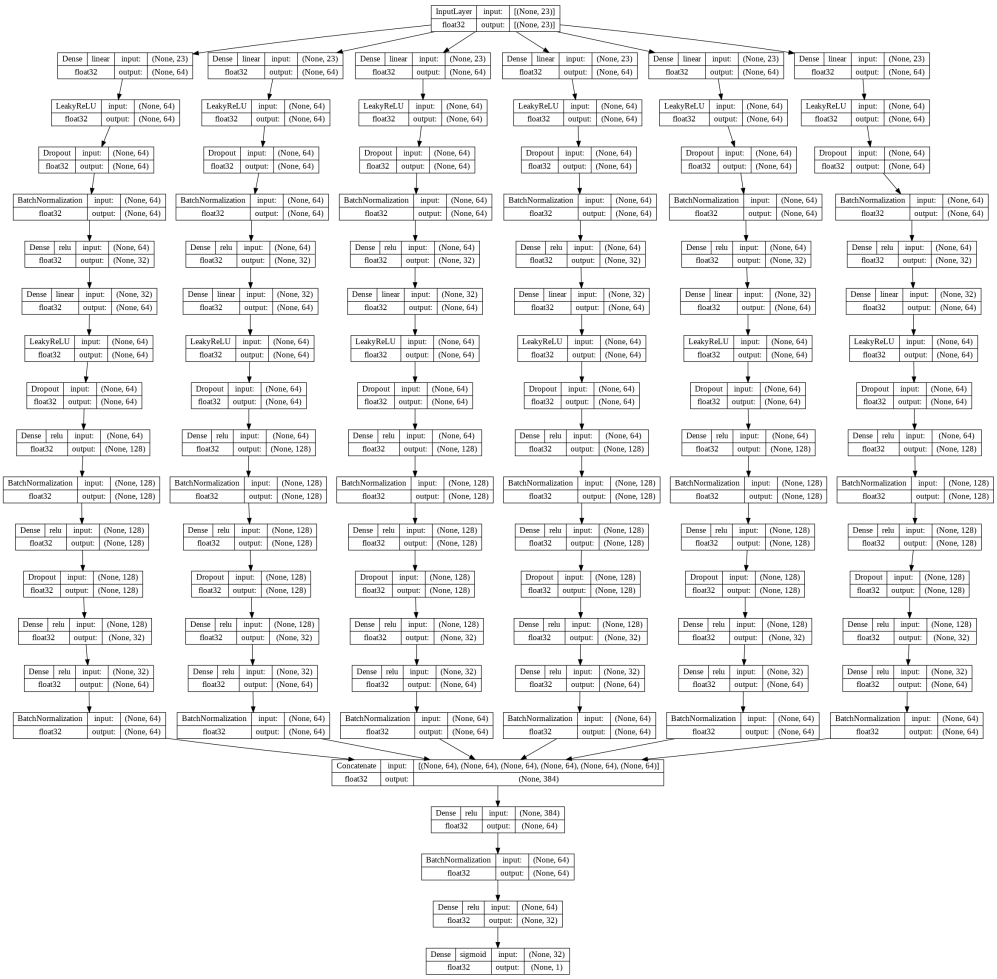
III. Implementation Details

1. Model Architecture (I tried XGBoost, CATBoost, Multi Layer Perceptron, Residual Multi Layer Perceptron)
   Below are structure of MLP and ResMLP (MLP gets the best performance among others)

   MLP



ResMLP



2. Hyperparameter & Data Processing & Results

   a. I only use the loading, attribute_n, mearsurement_n features (總共 23 個)

   b. For nan elements in dataset, I fill it with mean of the corresponding column.

   c. Model Comparison

Variable/ Model	MLP	ResidualMLP	CATBoost	XGBoost
Learning rate	4e-4	4e-4	1	1
Optimizer	Adam	Adam	X	X
Epochs	32	32	X	X
Batch size	32	32	X	X
Output dim	1	1	X	X
Max Depth	X	X	1	1
Training time	17min	25min	45 min	35min
N trees	X	X	104590	Default
Performance (Private Score) (No standardize)	.5908	.58014	.5431	.5412
Performance (Private Score) (standardize)	.5854	.5801	.5441	.546

3. Additional Information & Summary
   a. 此次 Project 若只輸出 0/ 1 類別，分數至多只有 0.504。
   b. 很明顯的 imbalanced data 用 oversampling 或 SMOTE 的效果不會很好，最後我是用 keras 裡的 class weight，根據每一類別資料筆數提升權重。
   c. 有些類別是 categorical，像 material_n 等，對其做 Standardize 的成果不管是 Tree based 的 model 或是 Neural Net 的效果都不會比較好；但就 nominal 的 column 而言，Standardization 在 NN 的效果不會比較好但在 Tree based 的效果會有所提升。
   d. 對於有 NaN 的 Datapoint 來說，我使用 mean 填空，效果比 Random 好。
   e. 就 Neural Net 而言，我使用了 20 幾層，拓展到 30 幾層反而會讓 Performance 降低。
   f. 在 Neural Net 前面幾層的 activation 用 LeakyRelu 會比單純 Relu 好，但若後面也都用 LeakyRelu，Performance 反而會降低。
   g. 64 或 128 個 epochs 來訓練 Neural Network 反而會 over fit，32 是不會 under-fit 的最佳數量。
   h. 總之，這個 competition 的 dataset 可說是 very dirty，若有更多時間或許可以做 data wishing，對 importance 不高的 data point 做替換，或是對 Material 的組合做研究；另外，通常這個 structured data tree-based model 會表現比較好，或許是容易 over-fit 所以用 neural network 的 generalization 比較好。

IV. Result

V. Github

https://github.com/jayisaking/111MLFinal.git