Capstone Final Report

Danlei Qian
December 2,2020

1. Introduction

The project is based on the Kaggle competition - Mechanisms of Action (MoA) Prediction, with the goal to "classify drugs based on their biological activity". Pharmaceutical drug discovery aims to identify certain proteins that are associated with a specific disease, and then to develop molecules that can target those proteins.

In short words, this is a multi-label classification problem. In this competition, I explored several machine learning algorithms and feature engineering methods. And the solution in this report is only based on my best solution from the leaderboard.

The link of this competition is as follows: https://www.kaggle.com/c/lish-moa/overview

2. Data Description

The datasets are tabular data, consisting in the familiar shape of train and test files. The training features (train_features.csv) and the targets (train_targets_scored.csv) are two separate files. Each row corresponds to a specific treatment. Features g- signify gene expression data, and c- signify cell viability data. cp_type indicates samples treated with a compound (cp_vehicle) or with a control perturbation (ctrl_vehicle); control perturbations have no MoAs; cp_time and cp_dose indicate treatment duration (24, 48, 72 hours) and dose (high or low).

The goal of this competition is to predict the train_targets_scored.csv class probabilities for the test data. And approximately 25% of test data is provided(test_features.csv) for the calculation of public leaderboard while the rest 75% of test data that is assigned to the private leaderboard.

In addition, an optional set of MoA targets (train_targets_nonscored.csv) is given, which we don't need to predict, but can be used for supplementary information. And anonymous drug_id is also provided only for train set.

3. Methods

3.1 Model Architecture Diagram

My best solution is the ensemble of 3 single: Two-Phase NN model, Two-Heads ResNet NN model and TabNet model, and each of them is assigned with the same weight. Figure1 shows the overview diagram of my best solutions.

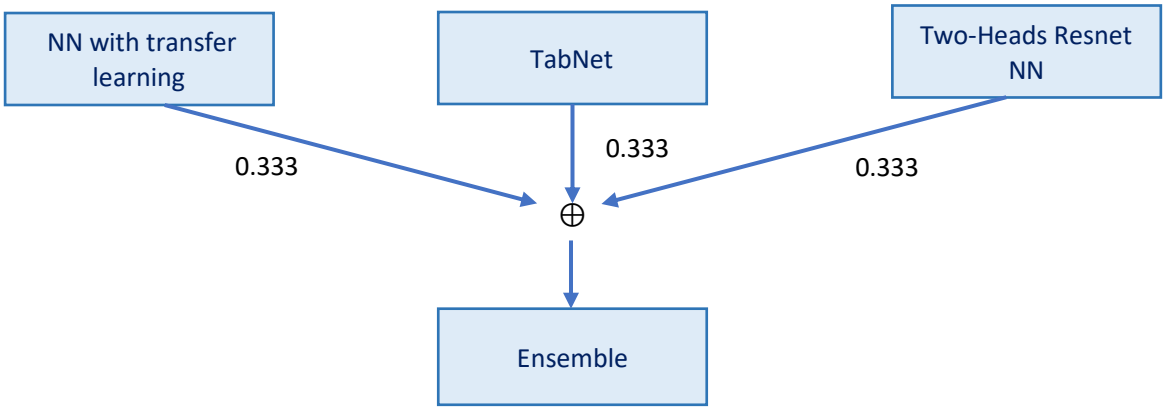


Figure1. Model Architecture Diagram

3.2 TabNet

TabNet is a Deep Neural Network for tabular data and was designed to learn in a similar way than decision tree-based models, in order to have their benefits: interpretability and sparse feature selection. Since 'TabNet inputs raw tabular data without any feature preprocessing', pointed out in the original paper, less processing steps are used for TabNet(as shown in Figure2). And the number of components of PCA are also reduced to 80 components for genes features and 10 components for cells features.

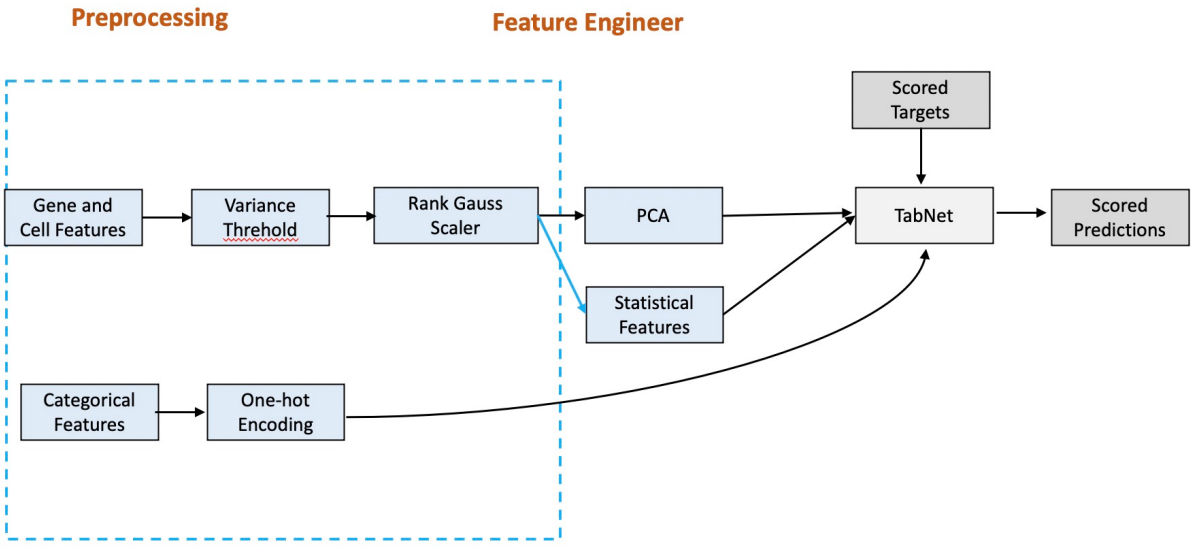
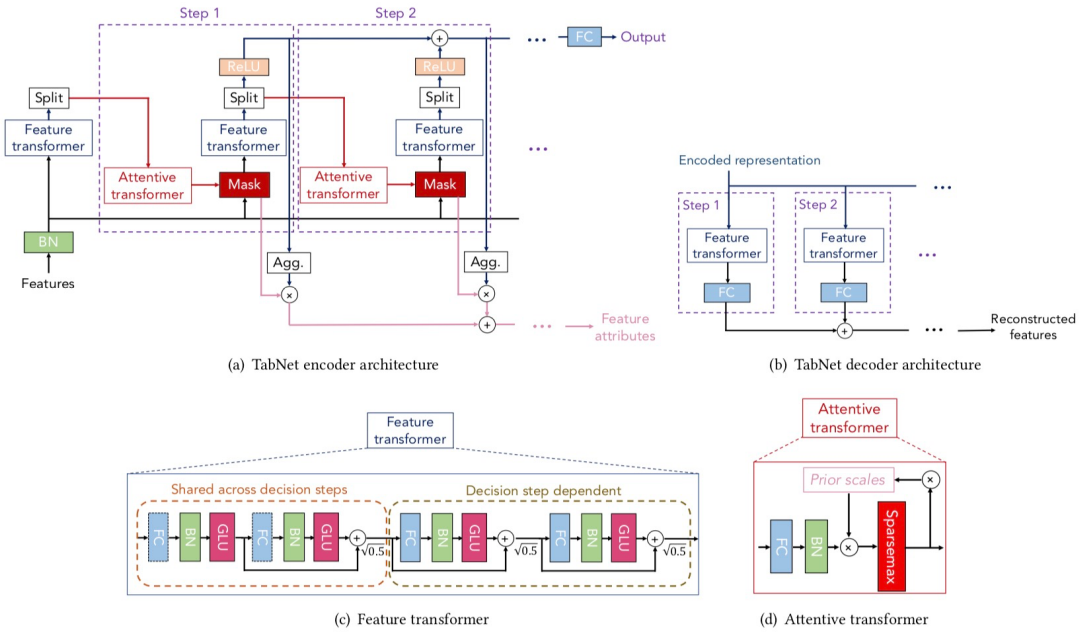


Figure2(a). TabNet Model



(a) TabNet encoder architecture

(b) TabNet decoder architecture

(c) Feature transformer

(d) Attentive transformer

Figure 4: (a) TabNet encoder for classification or regression, composed of a feature transformer, an attentive transformer and feature masking at each decision step. A split block divides the processed representation into two, to be used by the attentive transformer of the subsequent step as well as for constructing the overall output. At each decision step, the feature selection mask can provide interpretable information about the model's functionality, and the masks can be aggregated to obtain global feature important attribution. (b) TabNet decoder, composed of a feature transformer block at each step. (c) A feature transformer block example – 4-layer network is shown, where 2 of the blocks are shared across all decision steps and 2 are decision step-dependent. Each layer is composed of a fully-connected (FC) layer, BN and GLU nonlinearity. (d) An attentive transformer block example – a single layer mapping is modulated with a prior scale information which aggregates how much each feature has been used before the current decision step. Normalization of the coefficients is done using sparsemax [37] for sparse selection of the most salient features at each decision step.

Figure2(b). TabNet Model

3.3 Two-Phase Neural Network with Transfer Learning

This is my best single model, and model architecture can be seen from Figure5. The training process can be divided into two phases. In the first phase, the model is trained for non-scored targets by Adam optimizer for several epochs minimizing the binary cross entropy loss, with a learning rate of 1e-3, a batch size of 128, a minimum delta of 1e-5. And OneCycleLR learning rate scheduler is used with the maximum learning rate of 1e-3. Besides, the alpha value of label smoothing is set to 0.001.

In the second phase, the weights of learnt model from non-scored targets are being reused and transferred learning to scored targets. The weights except for the last layers (the FC5 module in figure4) are freeze for training again.

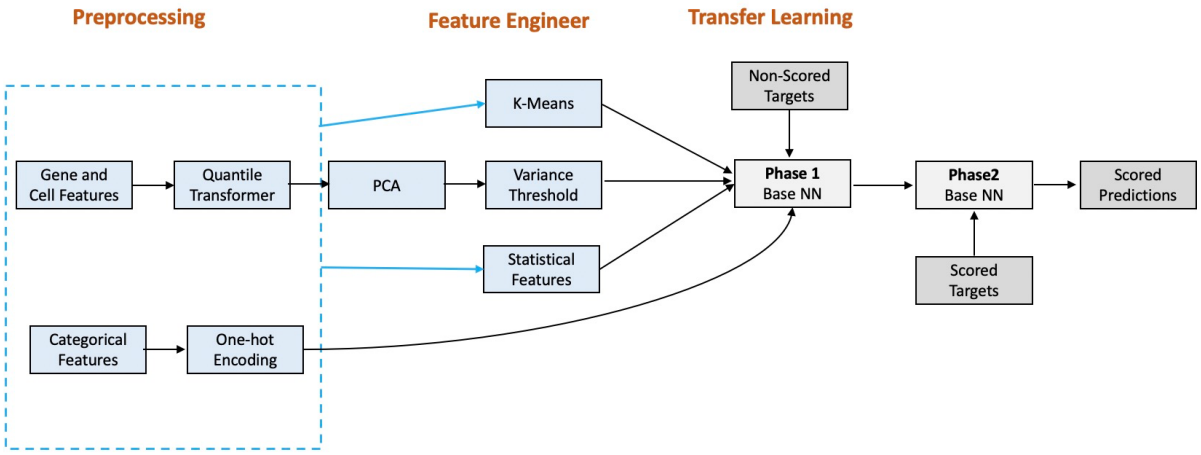


Figure3. 2-Phase NN with Transfer Learning

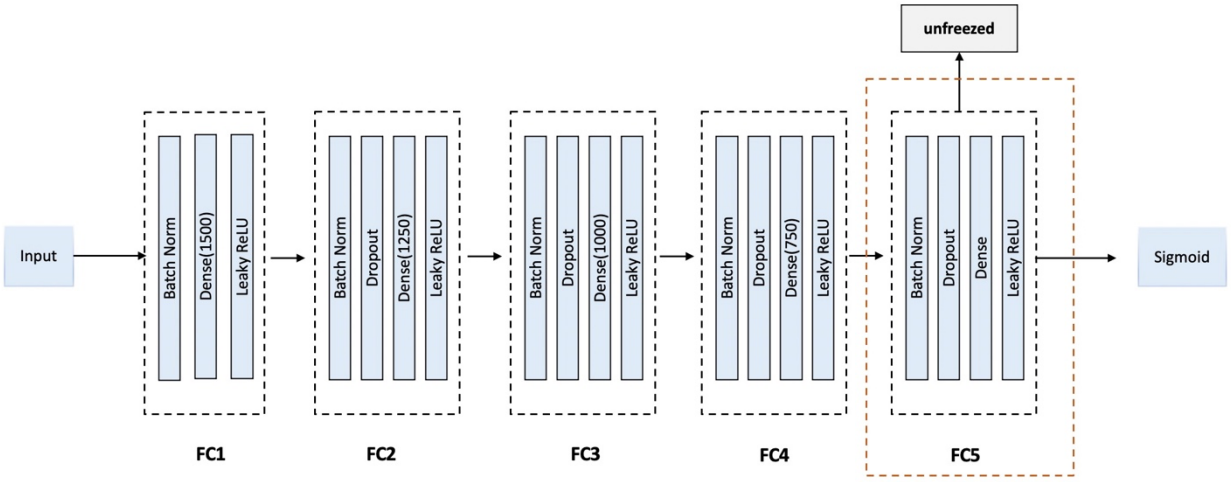


Figure4. 5-FC Base NN Block Topology

3.4. Two-Heads ResNet-like Neural Network

This model is based on ResNet-like NN with two inputs. Figure5 and Figure6 show the architecture of this model, and Figure7 give the detailed summary of this model. One input of this model is from genes and cells features after processing, while the other one is the 447 important features selected by t-test mentioned above.

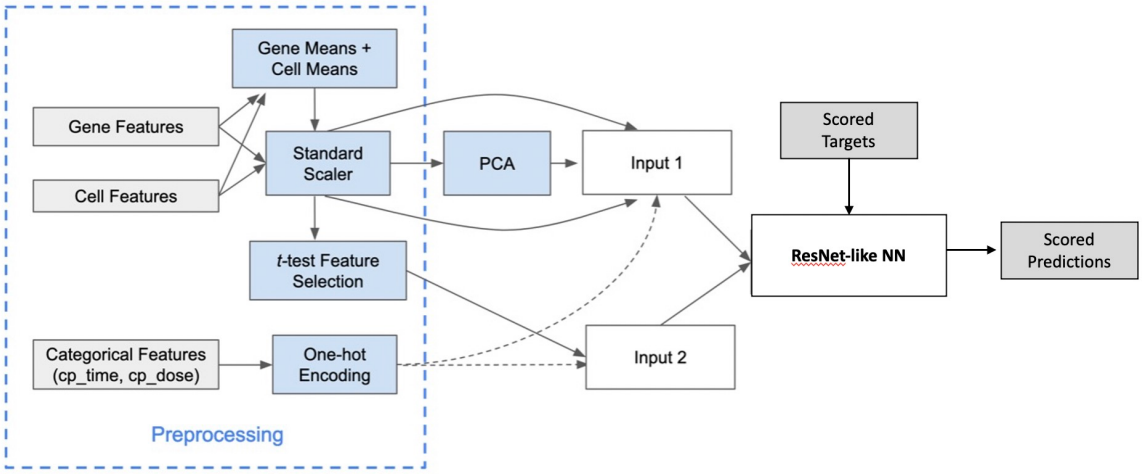


Figure5. 2-Heads ResNet-like NN

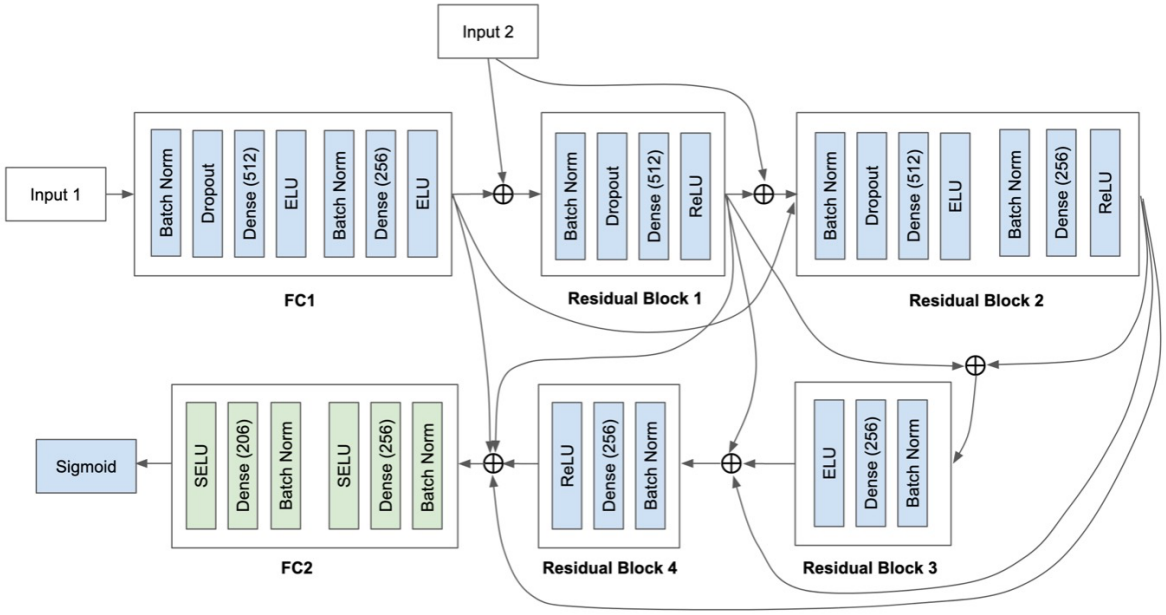


Figure6. 2-Heads ResNet-like NN Topology

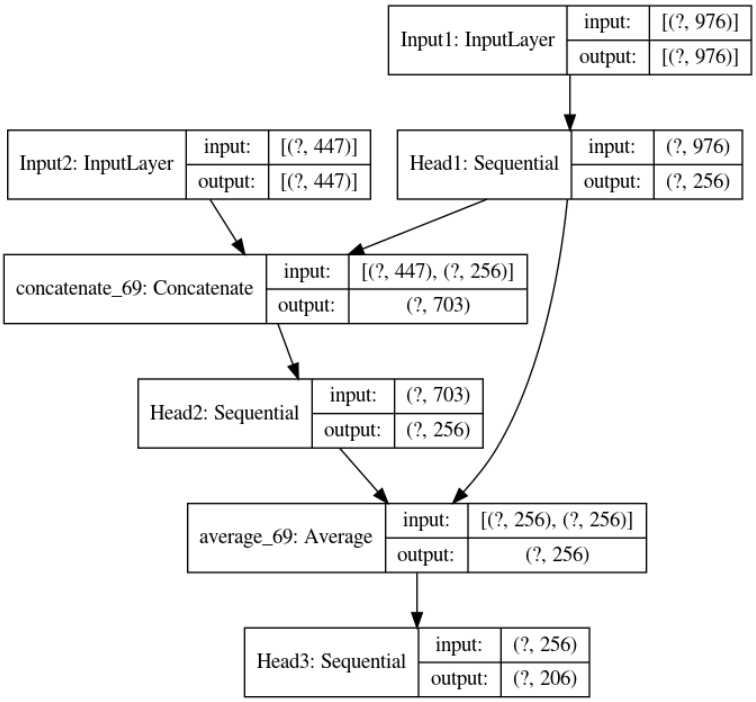


Figure7. 2-Heads ResNet-like NN Model Summary

3.5. Evaluation

Submissions are scored by the log loss:

$$\text{score} = -\frac{1}{M}\sum_{m=1}^{M}\frac{1}{N}\sum_{i=1}^{N}\left[y_{i,m}\log(\hat{y_{i,m}}) + (1-y_{i,m})\log(1-\hat{y_{i,m}})\right]$$

where:

- N is the number of `sig_id` observations in the test data ($i = 1, \ldots, N$)
- M is the number of scored MoA targets ($m = 1, \ldots, M$)
- $\hat{y_{i,m}}$ is the predicted probability of a positive `MoA` response for a `sig_id`
- $y_{i,m}$ is the ground truth, 1 for a positive response, 0 otherwise
- $log()$ is the natural (base e) logarithm

Note: the actual submitted predicted probabilities are replaced with $max(min(p, 1-10^{-15}), 10^{-15})$. A smaller log loss is better.

4. Experimental Setup

The Two-Phase NN model and TabNet model are implemented with Pytorch. The Resnet-like NN model is implemented with TensorFlow.

4.1 Data Preprocessing

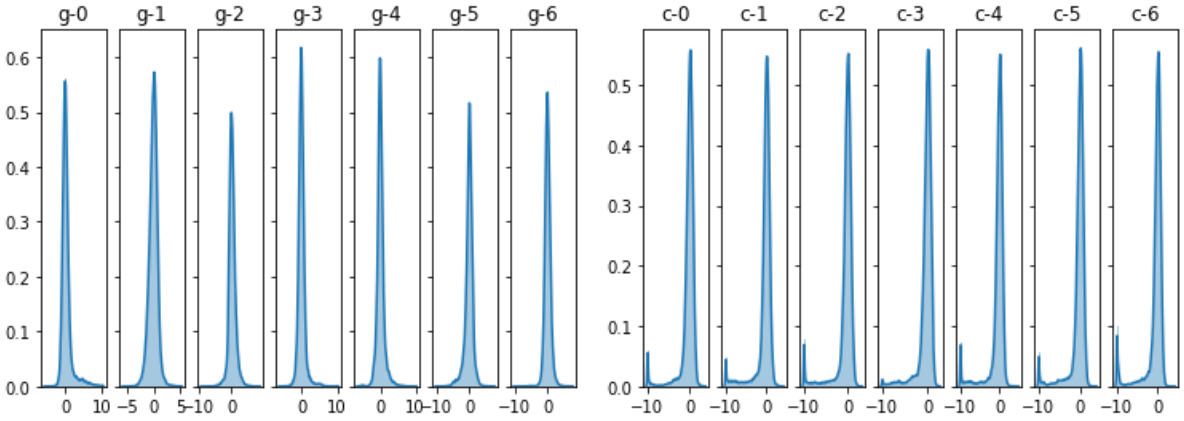


Figure7. (a) Features Distribution before Scaling

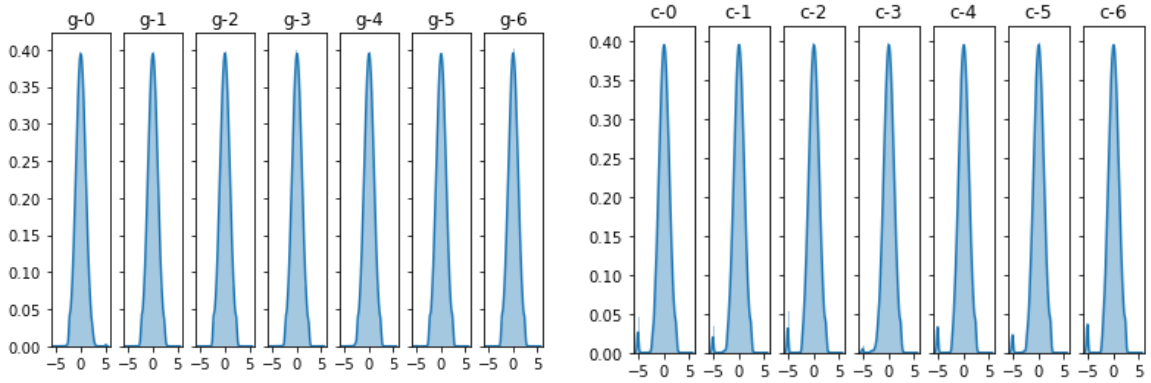


Figure7. (b) Features Distribution after Scaling

The main data preprocessing is scaling by using Quantile Transformation. As Figure7 shows, the raw data is spiky distribution rather than normal distribution, and after the transformation, the distribution of data become closer to the normal distribution, as intended. Besides, since there are three categorical features, one-hot encoding is also applied.

4.2 Feature Engineering

4.2.1 Dimensionality Reduction

To reduce the dimensionality of the datasets, principal component analysis (PCA) is applied, which is known as a linear transformation method that projects the data into another space, where vectors of projections are defined by variance of the data. The number of components is chosen based on the 95% of variance that can be explained. As we can see from the Figure8, 600 principal components of genes and 50 principal components of cells are needed to reach the 95% threshold.

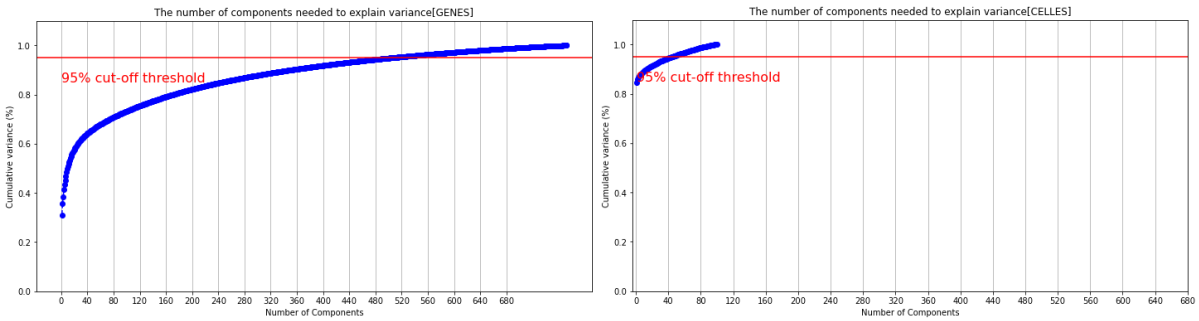


Figure8. (b) Features Distribution after Scaling

4.2.2 K-Means Clustering

K-means clustering is to aggregate data points together with certain similarities and discover underlying patterns. In this competition, K-Means is applied to increase more features. Silhouette Coefficient is used to decide the number of clusters. Its value ranges from -1 to 1 with 1 being the best and -1 being the worst.

As we can see from Figure9, silhouette score reach peaks at about 20 and 30 number of clusters for genes, while the score is very close to 1 with small number of clusters for cells.

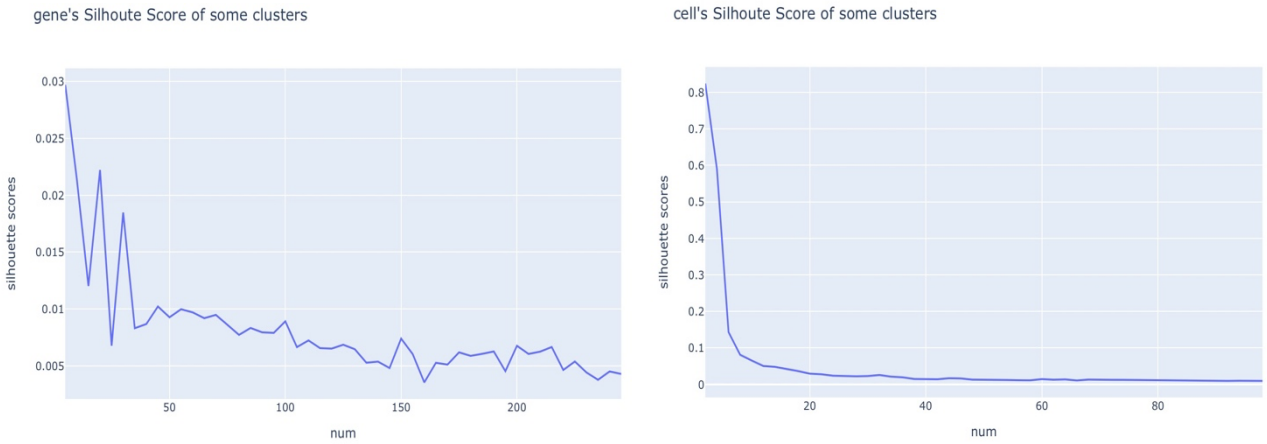


Figure9. Silhouette Score of Different Clusters

4.2.3 Correlation Analysis

As can be seen from Figure10, there are several high positive and negative correlations between some gene features, same as in cells features. When the threshold of correlation coefficient set to 0.9, one pair of gene features and 177 pairs of cell features are selected, which is too much. Thus, top 20 pairs of cell features with high correlation are finally selected. And the products of those pairs are added to be new features.

4.2.4 Feature Importance by T-test

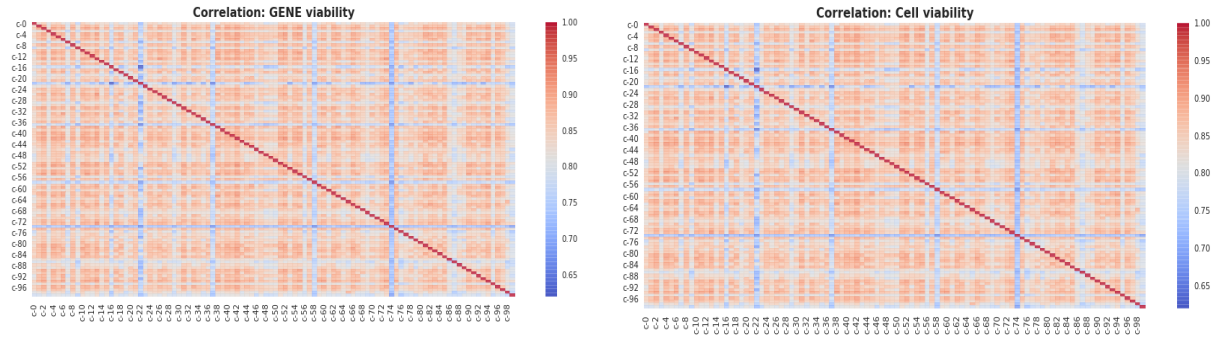


Figure10. Correlation Matrix of features

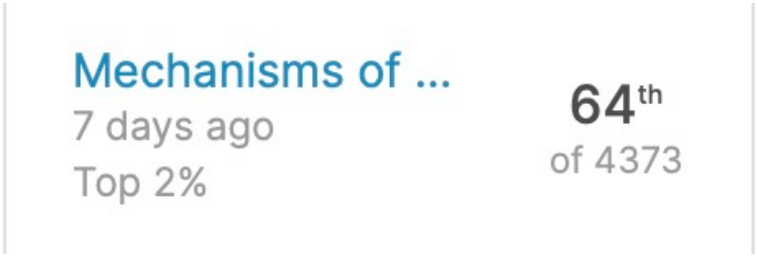
T-test helps find out 447 important features used in the 2-Heads ResNet model, which is applied by finding out the p-value for each target with each kind of gene or cell and then select the features whose 25% quantile of the p-values is smaller than the threshold of 0.01. The features are derived directly from the public notebook implemented by R.

5. Result

Single Model	**Seeds**	**K-folds**	**Cross Validation without Drug_id**	**Cross Validation with Drug_id**	**Public Score**	**Private Score**
Tabnet	1	10	0.016717		0.01841	0.01632
2-Phase NN With Transfer Learning	7	7		0.01563	0.01833	0.01623
2-Heads Resnet NN	7	10	0.01656		0.01850	0.01635
Ensemble with average weights					0.01824	0.01609

Table1. Model Scores

Table1 shows the scores of each model and the ensemble model. It is interesting that the simple average on the result of three models can bring significant improvements. And this ensemble model ranks 64th in this competition.



6. Conclusions and Limitations

In this competition, I've learnt that understating data is essential. In other words, digging into exploratory data analysis, and thus feature engineering would help us find more insights of modeling. Also, model diversity is very important in model ensemble.

At the same time, there are many limitations of my work in this competition that needs further study in the future. Firstly, I did not optimize the blending weights but just use the average weights. Secondly, I failed in implementing hyperparameter tuning by Optuna in this competition since the limitation of device and time. Thirdly, I did not implement self-supervised pretraining on TabNet introduced in the original paper. Besides, I only used three single models in my best solution with the limitation of 2-hour GPU running time for submission, while the winner used seven single models. Thus, it is also important to optimize the codes to speed up the running time.

Reference:

- K-means:

https://towardsdatascience.com/understanding-k-means-clustering-in-machine-learning-6a6e67336aa1

https://www.kaggle.com/yerramvarun/deciding-clusters-in-kmeans-silhouette-coeff

https://towardsdatascience.com/silhouette-coefficient-validating-clustering-techniques-e976bb81d10c

- PCA

https://www.mikulskibartosz.name/pca-how-to-choose-the-number-of-components/

https://www.kaggle.com/kushal1506/deciding-n-components-in-pca

- T-test

https://www.kaggle.com/demetrypascal/t-test-pca-rfe-logistic-regression#Select-only-important

- Adversarial Validation

https://towardsdatascience.com/adversarial-validation-ca69303543cd

- Label Smoothing

https://leimao.github.io/blog/Label-Smoothing

- TabNet

https://arxiv.org/abs/1908.07442

https://github.com/dreamquark-ai/tabnet

https://www.kaggle.com/marcusgawronsky/tabnet-in-tensorflow-2-0

https://www.kaggle.com/hiramcho/moa-tabnet-with-pca-rank-gauss

- ResNet-like NN

https://www.kaggle.com/demetrypascal/2heads-deep-resnets-pipeline-smoothing

https://www.kaggle.com/rahulsd91/moa-multi-input-resnet-model

- Transfer learning

https://www.kaggle.com/thehemen/pytorch-transfer-learning-with-k-folds-by-drug-ids

- Model Architecture Diagram and Topology

https://www.kaggle.com/c/lish-moa/discussion/201510