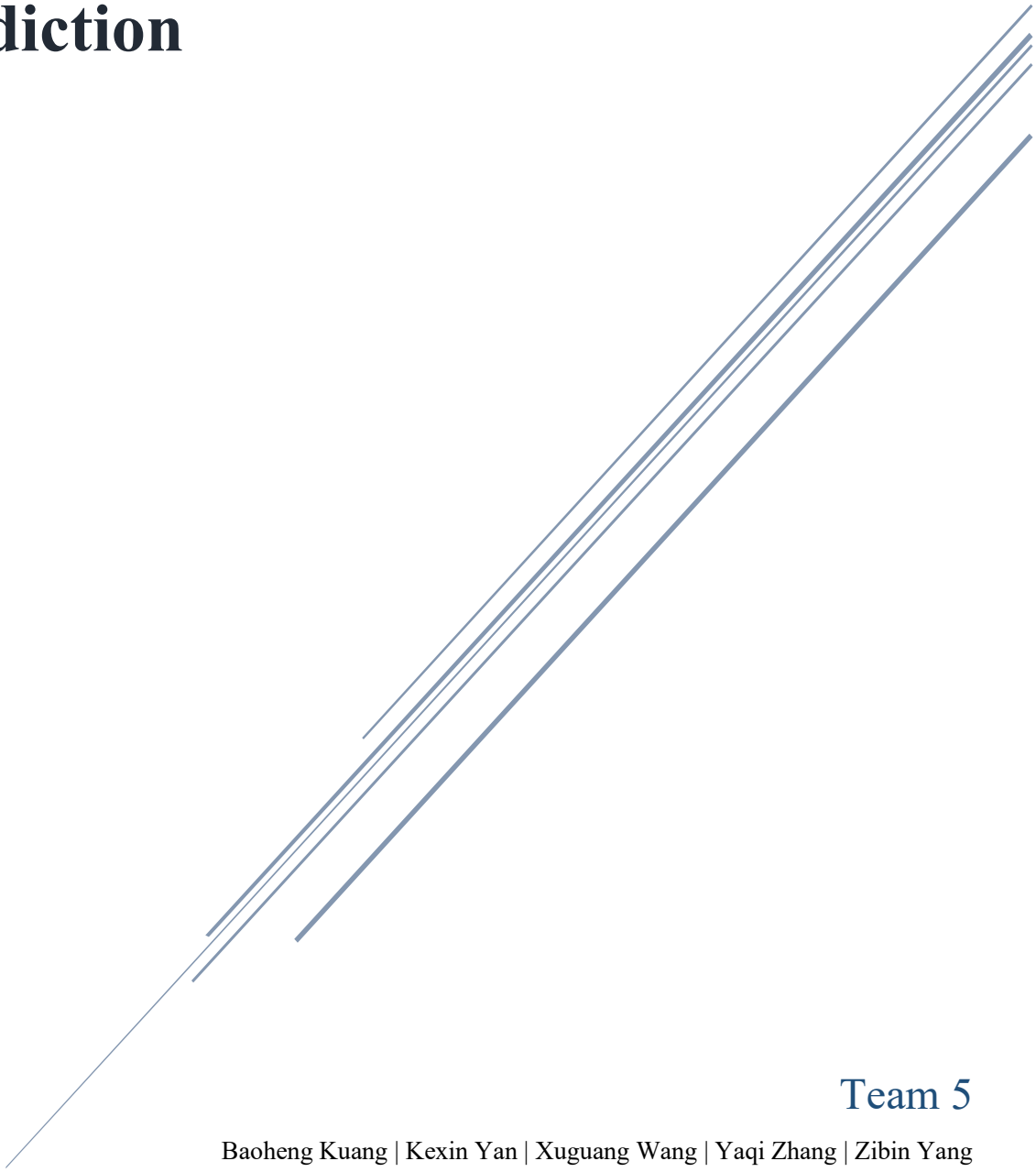
MIE 1628

Mechanisms of Action (MoA) Prediction



Team 5

Baoheng Kuang | Kexin Yan | Xuguang Wang | Yaqi Zhang | Zibin Yang

Contents

1. Introduction

1.1 Background

Mechanism of Action (MoA for short) describes the process by which a drug substance produces its pharmacological effect. It is such an important tool in drug discovery and characterization. With the study of MoAs, drug discovery has changed from the serendipitous approaches of the past to a more targeted model, based on an understanding of the underlying biological mechanism of a disease.[1] To determine the MoAs of a new drug, we can treat human cell samples with this drug, then use an algorithm to analyze the cell responses. And this algorithm can search for similarities with known patterns in large genomic databases, such as gene expression libraries or cell survival pattern libraries of knows MoAs drugs.[2]

In this project, our goal is to develop a model that can predict a compound's Mechanisms Action given its cellular signature, thus helping scientists advance the drug discovery process. We present histograms and heatmaps to explore the data, analyze and visualize the importance of the features, then conduct feature engineering and feature selection based on the data analysis. We will compare four different algorithms in predicting drugs' Mechanisms Action and tune the hyperparameters of the winner model, then present the model performance using log loss evaluation metric.

1.2 Data Description

The dataset containing gene expression and cell viability data, which is based on a technology that can simultaneously measure human cells' responses to drugs in a pool of 100 different cell types.[2] The data consists of 23814 samples with 875 features. The features can be divided into 5 types, gene expression data (g-), cell viability data (c-), treatment with a compound or control perturbation (cp_type), treatment duration (cp_time), and drug dose (cp_dose). The dataset has 206 binary MoA targets that are scored, and the predictions are represented as probabilities of activation for each of various proteins-targets. It is a multi-label classification problem and could be converted to binary classification problems for every 206 targets.

1.3 Evaluation Metric

The accuracy of solutions will be evaluated on the average value of the logarithmic loss function applied to each drug-MoA annotation pair, which is based on the requirements from Kaggle. The formula below shows how to calculate the accuracy on the average value of the log loss function applied to each drug-MoA annotation pair. We will use this evaluation metric in the model implementation and tuning part.

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

- N is the number of sig_id observations in the data
- M is the number of scored MoA targets
- $\hat{y}_{i,m}$ 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

2. Methods

2.1 Exploratory Data Analysis

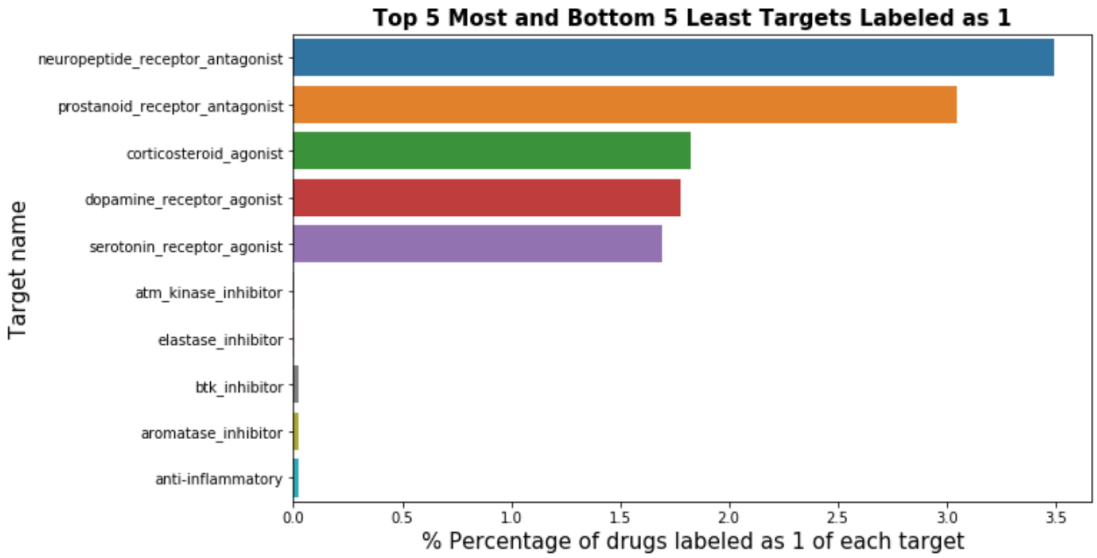


Figure 1 Top 5 and bottom 5 targets labeled as 1

For every 206 targets, each drug sample can be labeled 1 for a positive response or 0 otherwise. Figure 1 shows the top 5 and bottom 5 targets that contain the percentage of drugs labeled as 1. The highest target has 3.5% of drugs labeled as 1 while the lowest target only has less than 0.01% of drugs labeled as 1. Therefore, this data is highly imbalanced.

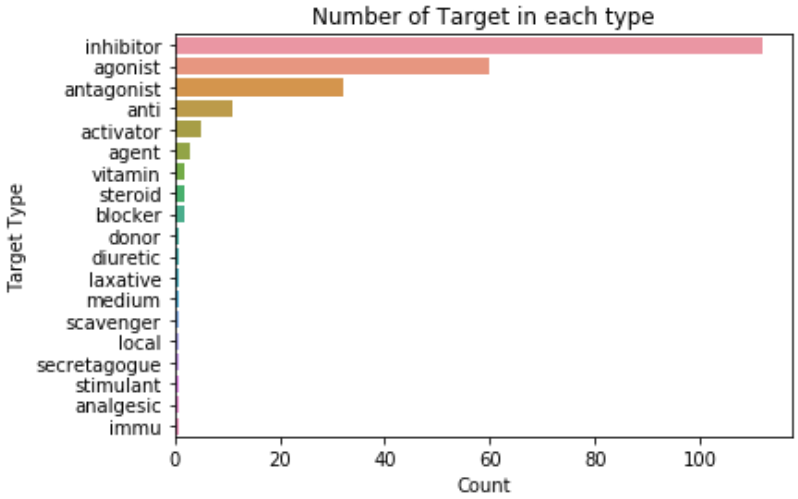


Figure 2 Number of targets in each type

The 206 targets are sorted into 19 functional groups as Figure 2 shown. The top 3 target types are inhibitor, agonist, and antagonist. In the model implementation and tuning step, we will take stratified samples based on these 19 groups.

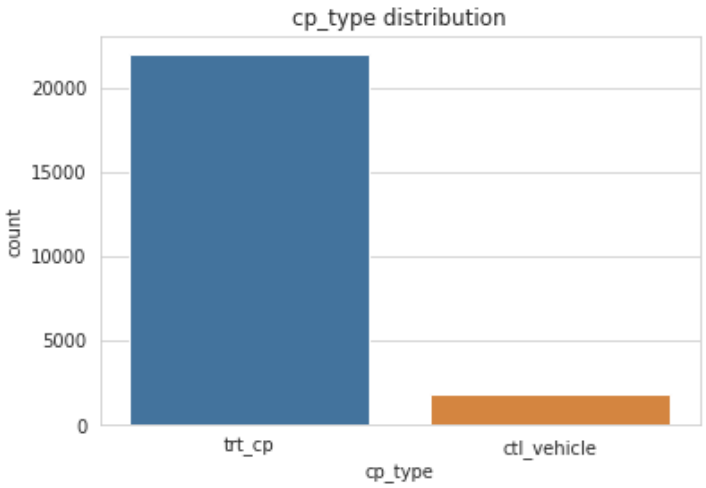


Figure 3 Number of targets in each type

In this dataset, three types of categorical features are observed, which are treatment duration (cp_time), drug dose (cp_dose), and treated with a compound or control perturbation (cp_type). Most of the samples are treated with drugs while only 7-8% of the samples are treated with control perturbation. Therefore, cp_type could be considered as an unbalanced feature, which may affect the prediction result.

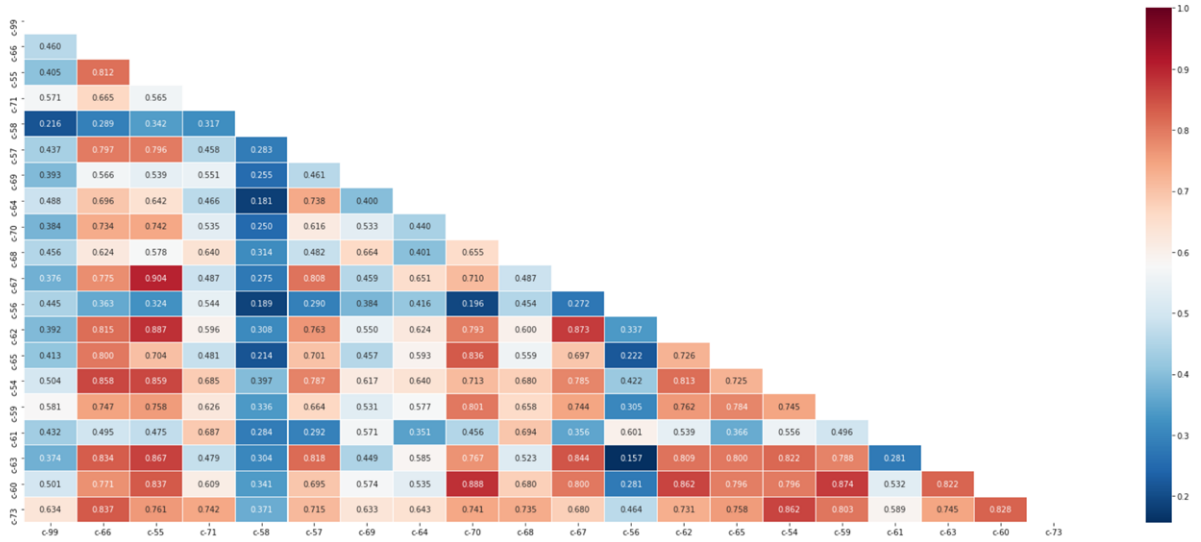


Figure 4 Correlations heatmap of cell viability features (Top 20)

There are 100 cell types of features related to cell viability. High negative cell viability values reflect a high number of cell deaths and low values high survival rates. The heatmap above shows correlations between cell viability features. This is the plot of the highest 20 correlations cell type features. Many high correlations between cell types features, which is something to be taken into consideration in feature engineering.

2.2 Data Processing

Because of the limited computing resource, based on the 19 functional groups, we take a stratified sampling of 10 targets (see Appendix A for target names) among the 206 targets to conduct the data processing and model implementation steps. For each target, we also leverage the stratified sampling technique to include 70% of the dataset in the train split and the rest in the test split. As mentioned above, the targets are very imbalanced. By using stratified sampling, we can guarantee that training and testing datasets have the same proportion of positive Mechanism of Action.

First of all, we work on data cleaning. We compute the total number of missing values in each feature and find that the MoA data is clean, so we have a conclusion that there are no missing values.

Secondly, we work on features engineering. As the information from the cell features heatmap in EDA, it shows the high correlation relationship within the cell features, therefore we decide to apply Principal component analysis (PCA) on these cell features. We apply 'StandardScaler' to standardize all the cell features first. Then, we use the 15 principal components from PCA to replace the 100 original cell features. Table 1 above shows the training and validation log loss from the Random Forest model with the hyperparameter as 50 trees and whether applying PCA.

Table 1 Log loss comparison between with and without PCA

PCA	Training Log Loss	Validation Log Loss
With	0.0115	0.0162
Without	0.0112	0.0161

Applying PCA has pros and cons. Although it drops some features and saves the model running time, it increases the log loss for both training and validation set. Therefore, we decide to move on without PCA since we cannot let the accuracy suffer.

Table 2 Log loss comparison between StringIndexer and Indicator variable

Encoding method	Training Log Loss	Validation Log Loss
StringIndexer	0.0114	0.0164
Indicator variable	0.0112	0.0161

Next, we move on to deal with three categorical features. For cp_time the treatment duration, there are three types of label which are 24, 48, and 72. For cp_dose the treatment dose, there are two types of label which are D2 and D1. Table 2 shows the training and validation log loss from the Random Forest model with the hyperparameter as 50 trees on 10 stratified samples with two encoding methods.

After comparing the log loss of two encoding methods which is done after RF has been selected to be our best model, we decide to convert two of the features, cp_time, and cp_dose into indicator variables with binary values of 0 and 1. As the number of unique values in these two features are only 3 and 2 respectively, our model won't suffer from a high number of features.

Table 3 Log loss comparison of whether dropping the low variance features

Low variance features	Training Log Loss	Validation Log Loss
With	0.0152	0.0187
Without	00152	0.0186

Regarding the feature selection, the cp_type indicates samples treated with a compound or with a control perturbation. Since the control perturbation has no MoA and has a small number

of observations in the training dataset, we decide to drop it. Also, by computing the variance of both cell and gene type features, we decide to drop features having a low variance with the threshold of 0.8, since these low variance features have low correlation to the prediction results. Having fewer features can increase the processing time. Table 3 shows the training and validation log loss from the Random Forest model with hyperparameter as 150 trees and whether dropping the low variance features on 20 targets.

2.3 Model implementation and Tuning

We select four machine learning models to perform the prediction of the Mechanism of Action. Because of the limitation of the Pyspark machine learning library, putting all the 206 targets within one model is not feasible. Instead, we build models for each of the targets and treat it as a binary classification problem. The four models are Logistic Regression, Random Forest, Gradient-Boosted Trees, and Multilayer Perceptron (Appendix B for hyperparameter settings). The Random Forest and Gradient-Boosted Trees are decision tree-based models, while the Multilayer Perceptron belongs to the feedforward artificial neural network. The training and validation log loss are shown in table 4 below:

Table 4 The training and validation log loss comparison of 4 models

	Logistic Regression	Random Forest	Gradient-Boosted Trees	Multilayer Perceptron
Training Log Loss	6.877076e-03	1.118801e-02	1.398806e-02	1.034532e-09
Validation Log Loss	0.026674	0.016093	0.021059	0.116455

Random Forest is selected as our optimal model, which has the lowest validation log loss. Although the Logistic Regression and Multilayer Perceptron have better performance in the training dataset, there is a big difference between the training and validation log loss. In this case, we believe the overfitting problem occurs in these two models.

We have around 800 features. In order to reduce the dimensionality of the data and shorten the training time of the model, we try to perform a feature selection. Thus, we look at the importance of all features generated by the random forest model to drop the insignificant features. In this step, we use the same 10 targets which are used in the step of model selection to compute the feature importance. We decide to use average importance to demonstrate the importance of one feature over 10 targets because in Pyspark we have to build 10 models for 10 binary targets. Figure 5 shows the top and bottom 10 features based on their importance values. We find g-0 is the most important feature, and cp_time_72 is the most unimportant feature. Although the importance of cp_time_72 is almost 0, the difference in the importance of these features is not too large. Also, the importance of features decreases gradually, and there is no obvious gap to split important and unimportant features. Therefore, we decide to not drop

any features.

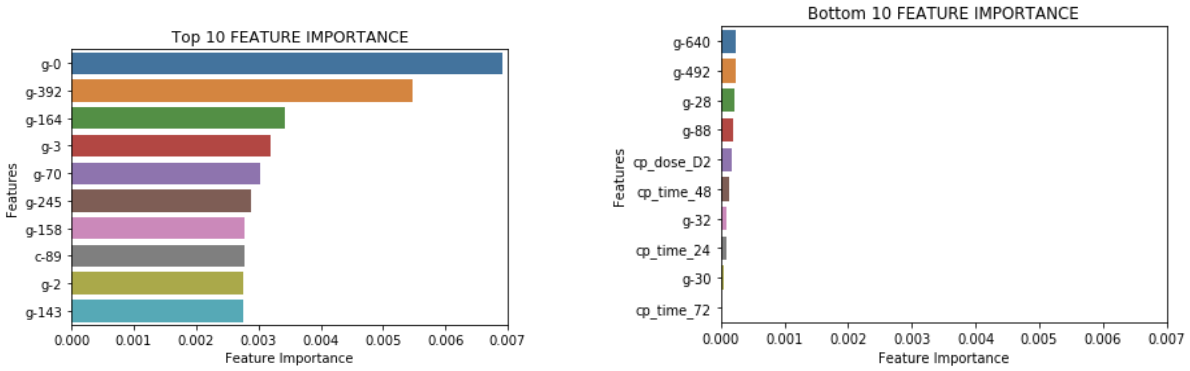


Figure 5 Top and bottom 10 feature importance

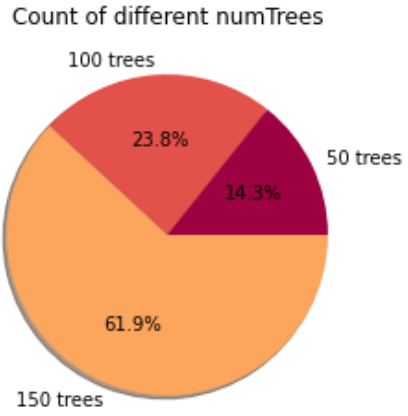


Figure 6 Model comparison for different number of trees

Next, we perform the hyperparameter tuning with cross-validation for the Random Forest. As Pyspark does not support Log Loss as a built-in evaluator, we implement our own Log Loss class, which is used in the cross-validation step. Due to the limitation on the computational power, we only tune the most important hyperparameter which is the number of trees on a subset of targets through cross-validation. The subset contains 20 targets which are selected by stratified sampling as well. Random forest models with 50, 100, and 150 trees are applied for each target. The function cross-validation uses 3 folders to choose the optimal model for each target. Thus, in total, after cross-validation, we have 20 models. As shown in figure 6, in 20 models, 61.9% of the models prefer to set the number of trees to 150, which means the random forest model with 150 trees outperforms other models for the most targets. Therefore, the final model we select is the random forest with 150 trees. Then we apply this model to the test set and find that on 205 targets, its mean test log loss is 0.0138. Moreover, there is no overfitting problem because its mean training log loss is only 0.0111. Besides, table 5 shows the top 10 targets with the lowest test log loss, we find 7 out of 10 targets are in inhibitor type so a conclusion that the model performs well in inhibitor type targets can be obtained.

Table 5 Top 10 targets with the lowest test log loss of final model

	Index	Target Name	Test LogLoss	Train LogLoss
0	70	coagulation_factor_inhibitor	0.000324	0.000327
1	40	autotaxin_inhibitor	0.000340	0.000272
2	125	lxr_agonist	0.000392	0.000426
3	165	protein_phosphatase_inhibitor	0.000394	0.000339
4	35	atm_kinase_inhibitor	0.000406	0.002134
5	76	diuretic	0.000575	0.000473
6	54	calcineurin_inhibitor	0.000594	0.000401
7	15	ampk_activator	0.000613	0.000711
8	196	tropomyosin_receptor_kinase_inhibitor	0.000615	0.000527
9	36	atp_synthase_inhibitor	0.000616	0.001261

3. Results and Discussions

Through the project, a random forest model is trained for predicting Mechanisms of Action (MoA) for drugs based on the gene and cell expressions. In the given dataset, 205 MoA are given as targets for each sample. The multi-label classification problem is broken into multiple binary classification problems. Each target is predicted based on the given features separately. The accuracy of the model is evaluated based on the average log loss of all binary classification models for all samples.

The random forest model was tuned with a 3-fold validation method. Due to the limitation on computational power, only 20 selected targets are used for tuning the number of trees in the random forest model. The tuning results show that the random forest model with 150 trees has the best performance. The average log loss in for the train set and test set were both low (0.0111 for the train set and 0.0138 for the test set). The log loss for the test set appeared to be slightly higher than the log loss for the train set for each sample and on average. Thus, the model is not subjected to overfitting or underfitting issues.

The model still can be improved in some aspects, if time and computational power are permitted. In the feature selection process, the recursive feature elimination method can be used to further reduced dimensionality. For training the model, an unsupervised method, such as the generative adversarial network, can be used for abandoning the dataset. Although there is still room for improvement, the performance of the current random forest model is fair and acceptable.

4. References

1. Corsello, S.M., Nagari, R.T., Spangler, R.D. et al. Discovering the anticancer potential of non-oncology drugs by systematic viability profiling. Nat Cancer 1, 235–248 (2020)

2. Mechanisms of Action (MoA) Prediction, https://www.kaggle.com/c/lish-moa/data

5. Contribution

Exploratory Data Analysis: Baoheng Kuang, Yaqi Zhang

Data Processing: Xuguang Wang, Zibin Yang

Logistic Regression: Xuguang Wang

Random Forest: Zibin Yang

Gradient-Boosted Tree： Kexin Yan

Multilayer Perceptron: Baoheng Kuang, Yaqi Zhang

Cross-Validation and Feature Importance: Kexin Yan, Zibin Yang

6. Appendixes

Appendix A

10 Target Names by Stratified Sampling:

aldehyde_dehydrogenase_inhibitor
fungal_squalene_epoxidase_inhibitor
nitric_oxide_synthase_inhibitor
p38_mapk_inhibitor
acat_inhibitor
casein_kinase_inhibitor
potassium_channel_antagonist
imidazoline_receptor_agonist
dopamine_receptor_antagonist
gonadotropin_receptor_agonist

Appendix B

Hyperparameter Settings:

Logistic Regression: maxIter = 10
Random Forest: numTrees = 50
Gradient-Boosted Trees: maxIter = 100, maxDepth = 5
Multilayer Perceptron: maxIter = 100, layers = [812, 128, 128, 2], blockSize = 128