Comparing Machine Learning Techniques for Predicting Cellular Responses to Chemical Perturbations

Bokai Lai, Charles Zhang

April 27th 2024

Abstract

This project explores the application of machine learning models to predict the differential expression of genes in human peripheral blood mononuclear cells (PBMCs) in response to various small molecule treatments. Utilizing a dataset provided by a Kaggle competition, we tested several models including Decision Tree, Random Forest, K-Nearest Neighbors (KNN), and Linear Support Vector Regression (SVR). Our evaluation focused on the models' ability to predict changes in Myeloid and B cells using Mean Root Relative Mean Squared Error (MRRMSE). The results showed that Linear SVR outperformed the other models, suggesting a linear relationship in the dataset that it could effectively leverage. Future work will explore more complex models and advanced tuning to enhance predictive performance. This study advances our understanding of how machine learning can be applied in pharmacogenomics to predict cellular responses to drugs, contributing to the development of novel therapeutics.

1 Introduction

Recent progress in single-cell technologies has significantly deepened our understanding of cellular functions and interactions. However, applying these technologies in drug discovery poses challenges due to the complexity of linking chemical disturbances to cell reactions. Traditional approaches are expensive and limited by the range of cell types suitable for high-throughput assays. Moreover, existing databases like the NIH-funded Connectivity Map (CMap) primarily focus on cancer cell lines and cover less than 5% of all genes, potentially providing an incomplete representation of general human biology.

This challenge originated from a Kaggle competition, prompting this project to fill the void in predictive modeling for the effects of drug perturbations across various cell types. The project's objective is to create a predictive model that harnesses a new single-cell perturbational dataset from the competition, featuring human peripheral blood mononuclear cells (PBMCs) treated with different small molecules. By using cells from healthy individuals and integrating multi-omic baseline data, we plan to predict differential expression levels for Myeloid and B cells for most compounds, thereby advancing the knowledge and development of novel therapeutics.

This project will showcase the use of various machine learning models to predict cellular reactions to chemical perturbations. We will evaluate each model's performance focusing on accuracy, efficiency, and generalizability across various cell types. This methodology not only determines the most effective model for predicting changes in gene expression but also enhances our understanding of the biological processes involved.

2 Dataset

2.1 General Description

Based on the data set description provided by the competition organizer, in the dataset, differential expression (DE) is determined by treating human peripheral blood mononuclear cells (PBMCs) with 144 selected compounds, followed by measuring the gene expression profiles in different cell types using scRNA sequencing. To calculate DE, raw gene expression counts from each cell type are pseudobulked, meaning counts from all cells of the same type within a well are summed. These pseudobulked counts are then analyzed using Limma. Based on the output of Limma, p-values are calculated based on the significance of model weights for each compound for each gene.

The task is to predict differential expression values for Myeloid and B cells for a majority of compounds.

2.2 Feature Selection

We use `de_train.parquet` as training data provided by the competition, it is an aggregated differential expression data in dense array format. For each cell example, it provide `cell_type` (cell type based on RNA expressions), `sm_name` (the primary name for the compound), `sm_lincs_id`, `SMILES` (simplified molecular-input line-entry system representation of the compound), `control` (boolean indicating whether this instance was used as a control) as features. In addition, it provide `A1BG`, `A1BG-AS1`, . . . , `ZZEF1` fields with differential expression values as **labels to predict**.

In the testing dataset `id_map.csv`, it contains only values for a number of `cell_type` / `sm_name` pairs, with labels unknown. It is model's job to fulfill labels with differential expression values.

By looking at the training data, it has been discovered that `sm_lincs_id`, `SMILES` are implied by `sm_name`, that is, any pairs of data samples with same `sm_name` also shares the same `sm_lincs_id` and `SMILES`, and vise versa. Since the testing dataset only provide `cell_type` and `sm_name`, it make sense to drop other features and only use these two features during training.

3 Methodology

3.1 Why Machine Learning?

The application of machine learning techniques in our study is primarily driven by the inherent complexity of associating hundreds to thousands of chemical components with cellular responses. Additionally, the human genome comprises approximately 20,000 genes, which significantly complicates the analysis of large datasets. Given the substantial computational demands and the intricate nature of the data, modern machine learning methodologies offer a viable solution to efficiently address these challenges. By leveraging these advanced techniques, we aim to unravel the complex interactions within biological systems and enhance our understanding of cellular processes.
While it is commonly recognized that many Kaggle competitions rely on intricate ensembles finely tuned to the test data, our project is particularly focused on investigating whether certain simple model types can match or closely approach top-level performance. Therefore, we will steer clear of using highly complex ensembles and primarily experiment with straightforward models. Below are models we explored and implementation details.

3.2 Model Selected and Description

Decision Tree Regressor: We rely on the `DecisionTreeRegressor` provided by `sklearn.tree`. The selection of the Decision Tree method for our analysis is primarily due to its widespread recognition and simplicity in implementation within the machine learning domain. Our specific choice, the DecisionTreeRegressor, necessitates the conversion of categorical features into numerical format, for which we employ a LabelEncoder. This particular model is well-suited to our needs as it effectively manages continuous variables and integrates seamlessly with differential equation-based methods, such as limma, for computing p-values. While the Decision Tree Regressor may not be the most optimal model for every aspect of our study, its ease of use and straightforward implementation make it a practical choice in balancing complexity and performance.

Random Forest: We use the `RandomForestRegressor` provided by `sklearn.ensemble`. The selection of the Random Forest method is strategically aimed at enhancing the capabilities of the simpler Decision Tree approach. Although Random Forest requires marginally more computational resources than a single Decision Tree, it significantly mitigates the risk of overfitting—a common limitation in simpler tree models. This method is particularly effective for handling large datasets with high dimensionality, like ours, and exhibits robustness against noisy data. Overall, Random Forest offers a more sophisticated analytical framework that increases the accuracy and generalizability of our results without compromising excessively on computational efficiency.

K-Nearest Neighbour: We use the `KNeighborsRegressor` provided by `sklearn.neighbors`. We opted for the KNN method based on its ability to predict target genes through differential equations by interpolating locally among the nearest neighbors within the training set. This approach leverages the spatial distribution of data points to infer genetic behaviors. To optimize model performance, we employed GridSearchCV, an exhaustive search technique, to identify the optimal number of neighbors (k) that results in the lowest validation error, ensuring precision in our predictive analysis.

Linear SVR: We use the `LinearSVR` Vidhya 2020 provided by `sklearn.svm`. We have selected Linear Support Vector Regression to address our research challenges due to its robust performance in high-dimensional spaces, where traditional regression methods often falter under the curse of dimensionality. Linear SVR not only navigates these complexities effectively but also offers considerable flexibility in model tuning. Key parameters such as C, the regularization parameter, and ϵ, which defines the margin of tolerance in the epsilon-insensitive loss function, can be meticulously adjusted to enhance the model's performance. This methodical approach ensures that our model remains efficient and reliable, even in the face of complex, high-dimensional data sets.

4 Results

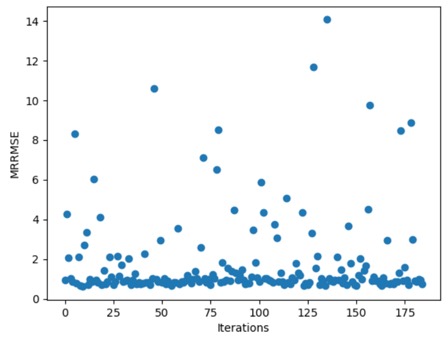
We will use the same error used by the original competition to evaluate our model's performances.

$$\text{MRRMSE} = \frac{1}{R}\sum_{i=1}^{R}\left(\frac{1}{n}\sum_{j=1}^{n}(y_{ij}-\widehat{y}_{ij})^2\right)^{1/2}$$

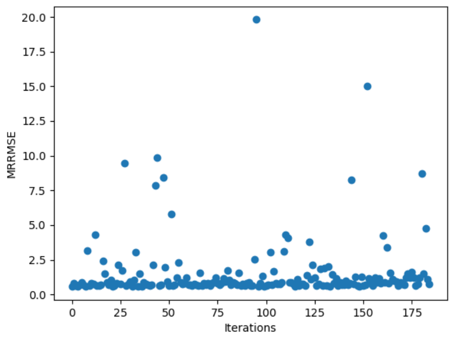
We have split the data into trainig and testing sets. Use training set to train the data, below is table of the MRRMSE for each models.

	Test Set	Private Set	Public Set
Decision Tree	1.816	1.762	2.291
Random Forest	1.524	1.417	1.877
KNN	1.469	0.832	1.092
Linear SVR	1.295	0.661	0.893

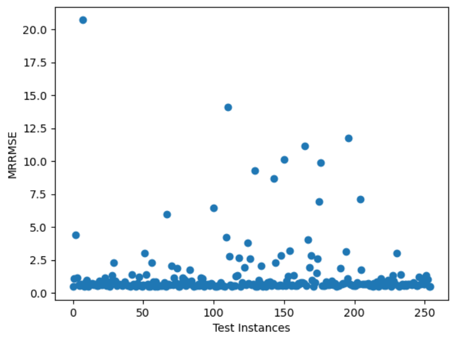
Table 1: Test Results



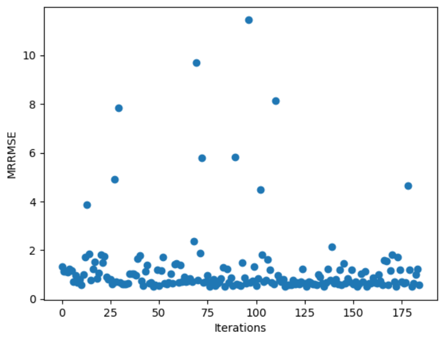
(a) Decision Tree



(b) Random Forest



(c) KNN



(d) Linear SVR

Figure 1: MRRMSE Plot

5 Discussion

5.1 Result Analysis

Based on the result above, we found that Linear SVR performed better than KNN regressor, Random Forest regressor, and Decision Tree regressor. In this section, we will discuss why that might be the case.

Linear SVR works well when there's a straightforward, linear relationship between the data's features and what you're trying to predict. This model is pretty robust against data that doesn't fit perfectly or is a bit unusual (outliers). Its top performance in our project suggests that our data probably followed a linear pattern, which Linear SVR could exploit more effectively than the other models.

KNN regressor, on the other hand, predicts outcomes based on what the nearest examples are doing. It can struggle if the data has lots of dimensions (features) or if the scale of the data isn't adjusted properly, which seems to have been a problem here. Since we used label-encoder to encode categorical data into 0/1 features, number of features may have exploded.

As for the Random Forest and Decision Tree regressors, these models build rules based on the data to make predictions. Decision Trees could overfit, which makes them perform poorly when faced with new data. Random Forest tries to fix this by using multiple trees and combining their results, which usually makes it better than using a single Decision Tree. However, if the trees make bad predictions, or if the data has a lot of unnecessary information, even Random Forest won't do very well.

Overall, Linear SVR's ability to handle linear relationships effectively, its robustness to outliers, and its suitability for high-dimensional data likely gave it the edge in our project compared to the other models, which struggled with these factors. This highlights the importance of choosing the right model based on what your data looks like and what you need from the analysis.

5.2 Future Work

Due to time constraints, we were unable to explore more sophisticated methodologies such as ensemble regression models and neural networks, which may offer enhanced predictive capabilities. Given additional time, we plan to investigate these complex models further and apply advanced hyper-parameter tuning techniques. These efforts aim to optimize our models' performance meticulously, ensuring that we leverage the full potential of modern machine learning technologies to address our research questions effectively.

References

Vidhya, Analytics (2020). *Support Vector Regression Tutorial for Machine Learning.* `https://www.analyticsvidhya.com/blog/2020/03/support-vector-regression-tutorial-for-machine-learning/`. Accessed: 2024-04-27.

A Supplementary material

This is the GitHub repo link where implementation of each method can be found:
`https://github.com/jasonllai/CPSC-440-Project`