Data Camp Project - Blackbull

Single-cell RNA-seq classification

Do Thanh Dat LE, Piseth KHENG - M2DS

December 13, 2023

1 Introduction

Nowadays, Single-cell RNA sequencing (scRNA-seq) is permitted to measure individual cell gene expression. This gene expression data would provide insights to distinguish cell types with different biological functions. In this data camp project, we aim to construct a pipeline with machine learning models to classify 4 cell types: Cancer cells, NK cells, T cells CD4+, T cells CD8+ using a subset of the scRNA-seq data from the scMARK benchmark dataset [4]. We present a supervised classification approach using machine learning models to accurately classify these 4 cell types based on gene expression data of individual cells. Our approach consists of quality control, data normalization, and feature selection to improve the performance of the classification pipeline. The performance of the pipeline is evaluated using balance accuracy. All the programming and computing will be done in Python.

2 Data Exploration

The data that we used is the subset of scMARK benchmark dataset with 1500 points split into 1000 training points and 500 test points. The data has 4 labels (cell types) for classification: `T_cells_CD8+`, `T_cells_CD4+`, `Cancer_cells`, `NK_cells`. The label proportion was shown in **Figure 1**.

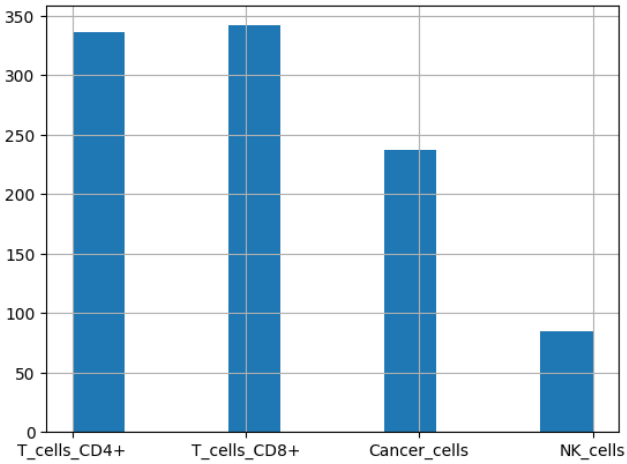

Figure 1: Label Proportion

Regarding the **Figure 1**, the labels indicates that the classes are imbalanced and there is no cell without labeling. The `T_cells_CD4+` and `T_cells_CD8+` accounts for the majority in the train data set, whereas cell `NK_cells` appears the least in the data set.

Moreover, we observed that each unique cell was described by 13551 features (genes), which is extremely high. High-dimensional data often contain a lot of noise which can negatively impact the performance of the prediction model. Therefore, it is necessary to reduce the dimension of the dataset in order to minimize noise, computing resources, and also enhance model performance.

The data is stored as a scipy sparse matrix in row format to save memory. However, many machine learning algorithms in scikit-learn could not work with this type, so we need to transform this data type into an array first.

A particularity of RNA-seq data is that total counts may vary widely between cells and/or genes. Therefore, we considered the distribution of total counts per cell and total counts per gene.

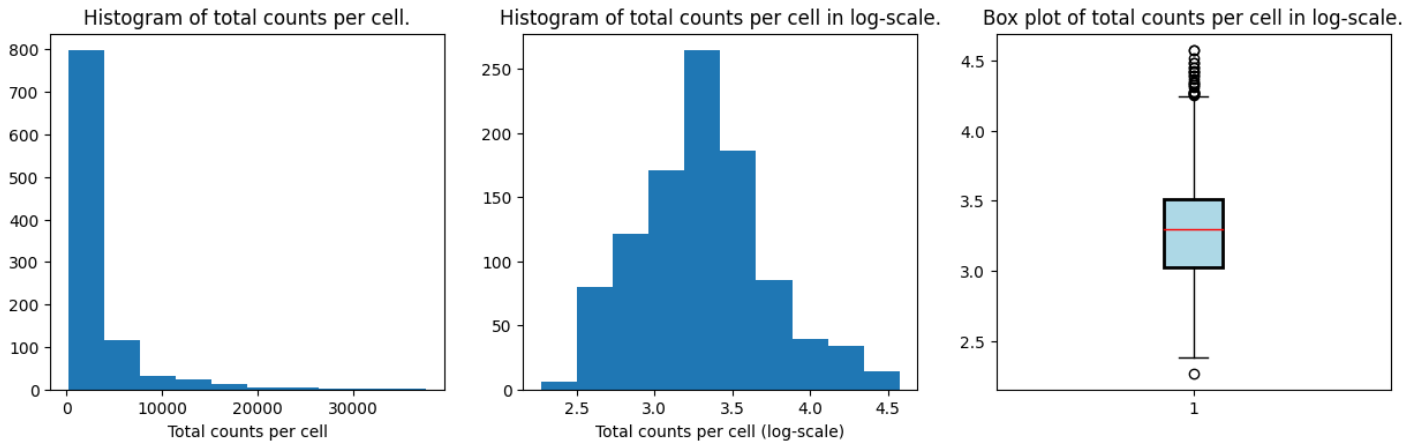

Figure 2: Total counts per cell distribution

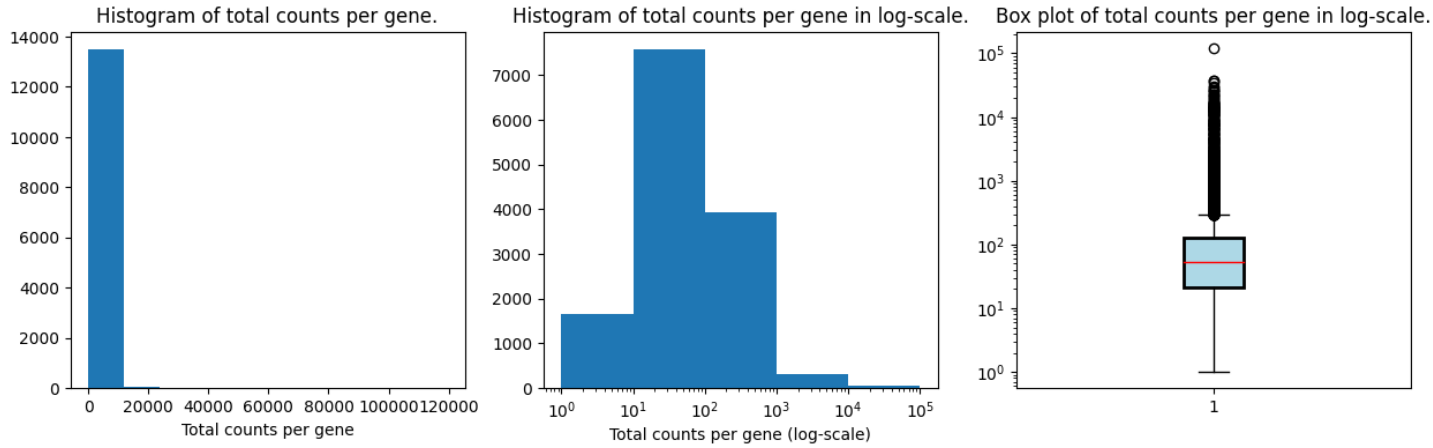

Figure 3: Total counts per gene distribution

We could see that in the **Figure 2** and **Figure 3**, the distribution of the total counts per cell and total counts per gene were right-skew distributions. These distributions suggest for some normalization of the counts. There are some genes that appear in very small quantities compared to other genes, as well as some genes that appear in very large quantities. Some cells have small number of genes compared to the others and somes have many genes. Therefore, we will do the quality control to filtering the initial data before normalization and prediction.

3 Data Preprocessing

In this section, we present our data preprocessing before training machine learning models. The preprocessing includes quality control, data normalization, and feature selection.

3.1 Quality Control

Before classification, we examined to remove poor quality cells to reduce noise in the dataset. A cell has a low total counts, a low number of expressed genes might have a broken membrane which can indicate a dying cell. Since these cells might distort our prediction, we removed them. To identify them, we need thresholds for the total counts and the number of expressed genes. We can set a fixed threshold, but it needs experience in biology. Therefore, we considered automatic thresholding via MAD (median absolute deviations), which is given by:

$$MAD = median(|X_i - median(X)|)$$

where X_i being the metrics (total counts, number of expressed genes, etc.) of each cell.

Reference to [3], we eliminate poor quality cells if they differ by 5 MADs which is a relatively permissive filtering strategy. In this case, because the distribution of total counts and number of expressed genes are not normal distribution so we removed based on the natural logarithm transformation of $(1+x)$ $(ln(1+x))$ where x is the total counts or number of expressed genes. Moreover, we also include 1 quality control `pct_counts_top20_genes`

in the **scanpy document**, which is the percentage of total counts that are attributed to the top 20 most highly expressed genes. After applying the quality control on the train data, we removed 9 cells with poor quality, so there were only 991 cells left in the train data.

3.2 Data Normalization

In order to normalize the train data, we conducted some normalization methods:

- Normalization by rows: we divided each count in each row (each cell) to the total counts in each row (total counts per cell)
- CPM (Counts per million): we did the Normalization by row first then we multiplied it to one million.
- Shifted logarithm (log transformation) with total count equal to the median of total counts

The Shifted logarithm is the normalization technique which is based on the delta method [Dorfman, 1938]. The delta method applies a nonlinear function to the raw counts and aims to make the variances across the dataset more similar. The shifted logarithm normalizes data by the following nonlinear function:

$$f(x) = log\left(\frac{x}{s} + x_0\right)$$

where x is the raw counts, s is a size factor and x_0 is a pseudo count.
The size factor for a cell can be calculated by

$$s = \frac{\text{total counts of a cell}}{L}$$

where L is the target sum.

In our project, we chose pseudo count $x_0 = 0$, L be the median of total counts per cell, and the log here is natural logarithm. Then, we obtained the distribution of total counts after normalization in **Figure 4**.

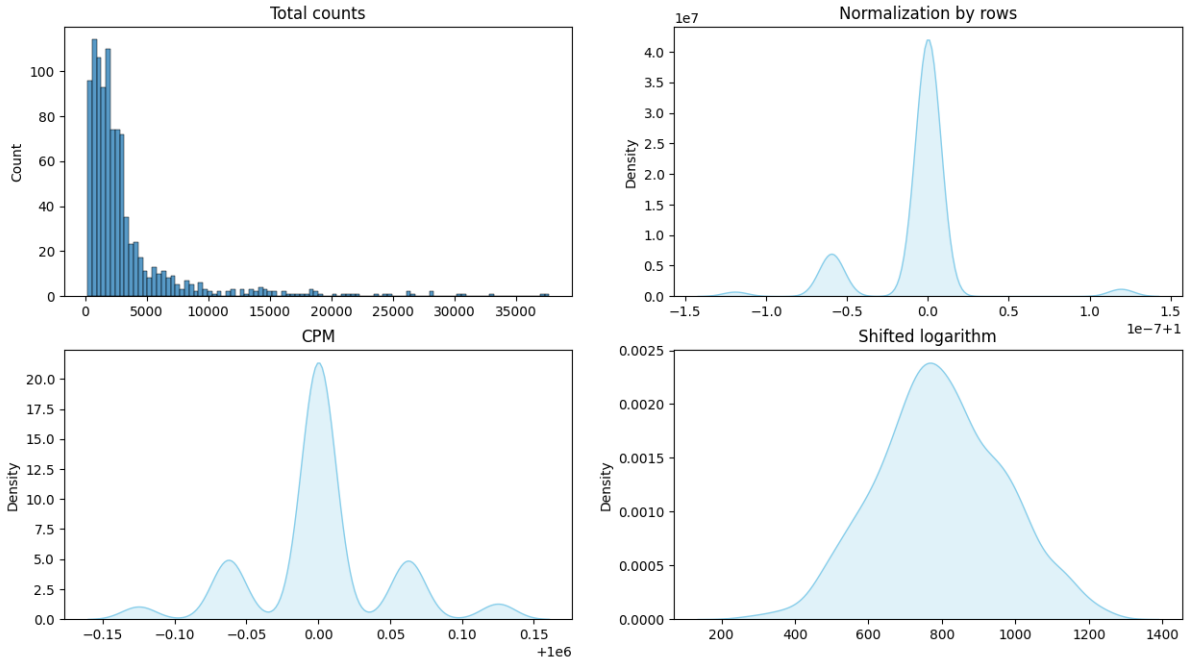

Figure 4: Total counts distribution after normalization

Regarding the density graphs in **Figure 4** after normalization, we decided to choose the Shifted logarithm to normalize the data since total counts after shifted logarithm normalization have a distribution most similar to the normal distribution.

3.3 Feature Selection

In our study, we observed that Single-cell RNA-seq data often contains many genes with zero counts, which are considered uninformative. To address this issue, we plan to filter out these uninformative genes from the dataset and focus only on the most useful genes for the downstream analysis.

3.3.1 Highly Deviance Genes

Previously, normalization involved Counts Per Million (CPM) and log transformation. However, when the data contains exact zeros, it becomes challenging to use log transformation. To overcome this problem, analysts often add a small pseudo count, such as 1 using log1p, before log transformation. Yet, the arbitrary choice of this count can introduce bias, which can impact subsequent analysis and gene feature selection. Low pseudo-count values near zero can amplify variance in genes with zero counts, as observed by Townes et al. in 2019.

To address this, Germain et al.[3] introduce a different strategy for choosing features by utilizing deviance directly on the original counts. This calculation of deviance helps determine if genes show a constant expression pattern across cells. Genes showing constant expression follow a multinomial null model, which is approximated using binomial deviance:

$$D_j = 2\sum_i \left[x_{ij} \log \frac{x_{ij}}{n_i \hat{\pi}_j} + (n_i - x_{ij}) \log \frac{(n_i - x_{ij})}{n_i(1 - \hat{\pi}_j)} \right]$$

where,

- D_j indicated the deviance of each gene.
- x_{ij} number of expressed gene-j in cell-i.
- $n_i \pi_j$ the fitted value.

Genes with high deviance values suggest they don't fit a constant expression pattern across cells, indicating their high informativeness. The method organizes genes by their deviance values and only selects those demonstrating significant deviance, presenting an alternative method to the conventional gene selection approaches in scRNA-seq analysis.

Since we couldn't use the binomial deviance function directly from the package, we had to interpret it locally based on the scanpy documentation[1]. We have selected several highly informative genes for the downstream analysis including the top 8000, and 4000.

3.3.2 Highly Variables Genes (HVGs)

Apart from the Highly Deviance Genes, we also tried another more simple feature selection method called Highly Variables Genes (HVGs)[2]. This method was referenced from (Lun, McCarthy, and Marioni 2016). We identified a set of top highly variable genes (HVGs), which could be used to define major cell types. This method was originally developed for single-cell RNA sequencing (scRNA-seq) data. The selection is based on the dispersion (seurat method) or the variance of expression level for each gene. The dispersion (seurat method) for each gene was calculated by

$$D = \frac{\sigma}{\mu}$$

where D is the dispersion for each gene, σ is the variance of expression level for each gene, and μ is the mean of expression level for each gene.

In this project, we only selected genes based on the variance of expression level for each gene. First, we identified the top genes with the most variance of expression level since the genes with higher variance in expression level will contain more information for classifying cell types than genes with low variance in expression level. Then, we chose the top 20% genes with the highest variance in expression level.

4 Modelisation and Evaluation techniques

In this process, we conducted two types of experiments where we tested the performance of the training data with the two types of feature selection methods, which are Highly Deviance Genes and Highly Variables Genes. We applied the feature selection on the train data and we only keep the selected genes on the test data.

We construct the pipelines with various models such as Linear models, Naive Bayes, K-nearest neighbors, Support vector machine, Multiclass classification, Decision tree, Ensemble method, Boosting model, and Neural Network. Due to the imbalanced classes of the data, we evaluated our pipelines based on a balanced accuracy score. Additionally, we evaluated the performance of these pipelines using the following steps:

- Firstly, we start by dividing the initial training set into two subsets: a training set (the set that is truly used to fit the model) and a validating set. The first set is used for training and the second set is for precision measurement. The training set had 80% of the initial data while the validating set had the remaining 20%.

- Then we created a data preprocessing function consisting of quality control, total counts normalization, and feature selection. Then we applied the preprocessing on the train data, remarking that we did not apply the quality control on the test set (only normalization and kept the selected genes from feature selection). We also included the preprocessing function in the pipeline.
- After that, we use the cross-validation with 5 folds to measure the performance of each model. Additionally, we plan to choose the best models for combination in the stacking model to enhance prediction performance.
- Finally, we select several best models and create the pipelines to predict on the testing dataset and those with the highly balanced accuracy score will be selected for the experiment on the local ramp-test before submitting to RAMP.

5 Results

In this section, we presented the balance accuracy, the train time, and the validation time of the pipelines that we tried during our project.

5.1 Highly Deviance Genes results

We conducted the feature selection with highly deviance genes to select the top 8000 and 4000 for the training process. After we finished fitting the models, we then used the 5-fold cross-validation to compare the models' performance using balanced accuracy with non-scaled data and scaled data. However, with the scaled data, there is a small improvement in some models' performance, but overall, there was no significant change.

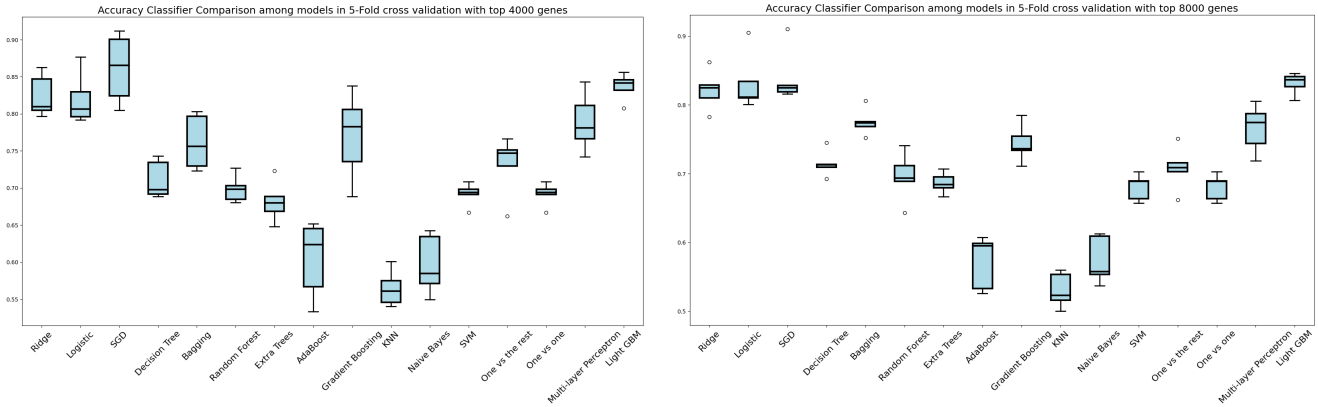

Figure 5: Balanced Accuracy Classifier Comparison among models in 5-Fold cross-validation with non-scaled data

After analyzing the boxplots (figure5) with the top 8000 and 4000 genes, we found that Ridge Classifier, Logistic Regression, SGD Classifier, Gradient Boosting, Multi-layer Perceptron (MLP), and Light GBM all had highly balanced classification scores. Consequently, we attempt to use these models with the top 4000 genes for testing on RAMP. Their performances are shown in **Table1**.

Pipeline	Top 4000 genes		
	Balanced accuracy	Train time	Validation time
Light GBM	0.85	39.466069	1.365933
SGD	0.84	4.292375	1.368014
Logistic Regression	0.83	12.718655	1.576035
MLP	0.83	20.436646	1.551991
Gradient Boosting	0.80	14.031128	1.276431

Table 1: Results of pipelines on RAMP

We observed that the pipeline utilizing the Light GBM model and selecting the top 4000 genes shows the highest balanced accuracy of 0.85 compared to other models and has a relatively low training time of 39.46 seconds and validation time of 1.36 seconds.

As we mentioned earlier, we will select the most potential models to combine in the stacking model to improve the prediction performance on the testing data. We tried various combinations of models as estimators for the stacking model on RAMP. Finally, we achieved the highest score (0.89) on RAMP with the following models:

- Stacking model with the top 8000 genes using base estimators include Ridge, SGD, MLP classifier, Extra Trees, Gradient Boosting, and Light GBM. The final estimator is Logistics regression. With this model, we achieved a balanced accuracy of 0.89, train time: 4892.345831, validation time: 2.889698. However, the training time is quite high.

- For the second stacking model with the top 4000 genes using base estimators include Ridge, SGD, MLP classifier, and Light GBM. The final estimator is Logistics regression. With this model, we achieved a balanced accuracy of 0.88, train time of 209.809646, and validation time of 1.877203. The training time of this model is faster than the first one.

Nonetheless, we will test these two stacking models using local data to evaluate their predictive performance. Regarding the figure 6, we can see that both models have shown a significant performance on the training data. Looking at the confusion matrix, we can see that both models do a good job of predicting Cancer_cells and NK_cells. However, the stack model with the top 8000 genes performs better than the other model when it comes to predicting T_cells_CD4+ and T_cells_CD8+.

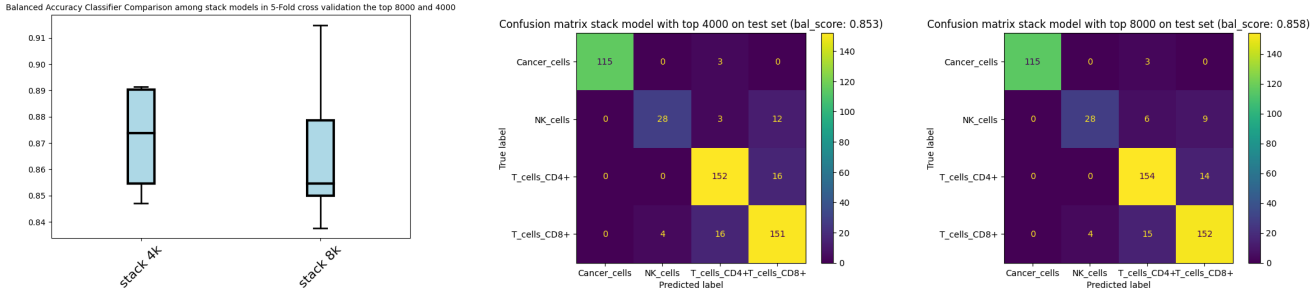

Figure 6: Stacking model results and confusion matrix on local test

Both models have similar prediction performance. However, the stacking model that utilizes the top 4000 genes has a significantly faster training time compared to the other model. Despite this, we prefer the stacking model with a higher balanced accuracy score (stacking model with the top 8000), even if it takes longer to train. Though the training time is higher, it is still acceptable.

5.2 Highly Variables Genes results

We used Highly Variables Genes method to select top 10% genes and top 20% genes. Then, we fitted the models, we then used the 5-fold cross-validation to compare the models' performance using balanced accuracy with non-scaled data and scaled data. We seen that the scaled data was seem to be better so we chose to scale the data. The results for scaled dtata were illustrated in **Figure 7**

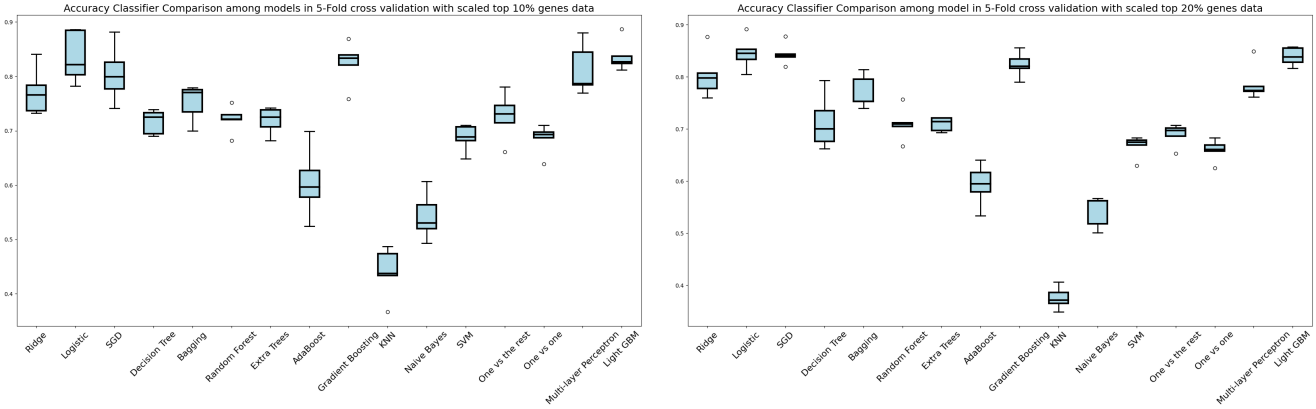

Figure 7: Balanced Accuracy Classifier Comparison among models in 5-Fold cross-validation with scaled data

Regarding the boxplots, there are several models that have potential in classification: LightGBM, Logistic Regression, SGD, Gradient Boosting, Ridge Classifier, and Multi-Layer Perceptron (MLP). We will create pipelines for these models. The results of those pipelines were presented in **Table 2**

Pipeline	Top 10% genes			Top 20% genes		
	Balance accuracy	Train time	Validation time	Balance accuracy	Train time	Validation time
Logistic Regression	0.85	12.543973	1.043762	0.85	26.685676	1.167990
Light GBM	0.85	28.222524	1.172846	0.85	38.053162	1.361378
Gradient Boosting	0.85	344.285727	1.115084	0.84	497.091907	1.258523
SGD	0.77	1.929863	1.143406	0.79	2.397393	1.313304
MLP	0.82	7.703455	1.510690	0.79	7.849618	1.406347

Table 2: Results of pipelines on RAMP with HVGs

We could see in the **Table 2** that the pipeline with Logistic Regression for the selected top 10% genes is the best in balance accuracy (0.85) as well as train time (12.543973) and validation time (1.043762). The LightGBM and Gradient Boosting are also good with the same balance accuracy (0.85). However, the train time and validation time of Gradient Boosting are very high compared to other models.

In order to improve the prediction performance, we will combine all those potential model above in the stacking model. We tried various combinations of models as estimators for the stacking model on RAMP. Finally, we also achieved the highest score (0.89) on RAMP by trying the following models:

- Stacking model with the HVGs top 20% genes using base estimators include Ridge, SGD, MLP classifier, Extra Trees, Gradient Boosting, and Light GBM. The final estimator is Logistics regression. With this model, we achieved a balanced accuracy of 0.89, train time of 310.191439, validation time of 2.234147. The train time was decreased significantly since we modified some parameters in the Gradient Boosting in the Stacking model, especially the parameter max_features = 'log2' to reduce the train time.

- For the second stacking model with the HVGs top 10% genes using the same base estimators include Ridge, SGD, MLP classifier, Extra Trees, Gradient Boosting, and Light GBM. The final estimator is Logistics regression. With this model, we achieved a balanced accuracy of 0.87, train time of 270.212576, validation time of 1.948987. The balanced accuracy decreased compare to the one with the HVGs top 20% genes. It could be because we loss some information by only seleted top 10% instead of 20%.

Nonetheless, we will also test these two models using local data to evaluate their predictive performance. Regarding the figure 8, we can see that both models have shown good performance on the training data, but the Stacking model with HVGs top 20% genes was superior in balance accuracy compared to the Stacking model with HVGs top 10% genes. If we look at the confusion matrix, we also see that the Stacking model with HVGs top 20% genes has a better performance than the Stacking model with HVGs top 10% genes when predicting NK_cells and T_cells_CD8+.

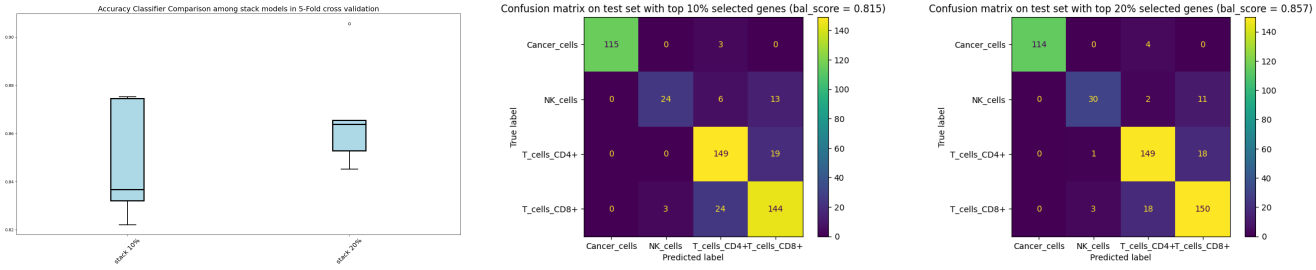

Figure 8: Stacking model results and confusion matrix on local test with HVGs

6 Conclusions

In conclusion, both feature selection methods have demonstrated a significant impact on data dimension reduction. This has led to reduced computing time for complex models and significant improvements in model performance. The Stacking model with base estimators including Ridge, SGD, MLP classifier, Extra Trees, Gradient Boosting, Light GBM, and Logistic Regression as a final estimator is the best model for Highly Deviance Genes 8000 genes and Highly Variable Genes top 20% genes. The best balance accuracy we achieved is 0.89 on RAMP. However, the train time is high caused by the Gradient Boosting, so we modify the parameters max_features = 'log2' of Gradient Boosting in the Stacking model with HVGs top 20% genes, and the train time decreased significantly to 310.191439.

References

[1] Scanpy document, computation of highly deviant genes. https://github.com/ktpolanski/scanpy/blob/deviantgenes/scanpy/preprocessing/_highly_deviant_genes.py.

[2] S. A. T.-P. M. V. C. Jantarika Kumar Arora, Anunya Opasawatchai. Computational workflow for investigating highly variable genes in single-cell rna-seq across multiple time points and cell types. *STAR Protoc.*, 4(3), 2023.

[3] A. S. Pierre-Luc Germain and M. D. Robinson. pipecomp, a general framework for the evaluation of computational pipelines, reveals performant single cell rna-seq preprocessing tools. *Genome Biology*, 21(1):227, 2020.

[4] O. F. J. J. D.-M. S. C. Swechha, Dylan Mendonca. scmark an 'mnist' like benchmark to evaluate and optimize models for unifying scrna data. *bioRxiv*, 2021.