

report

2024-12-09

Merge of both tasks' reports

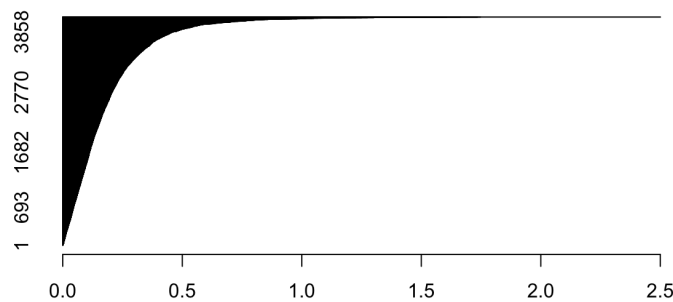
Task 1: Binary Classification

Abstract Task 1

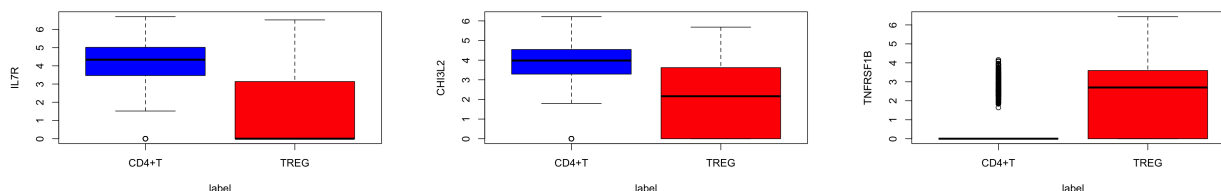
In the following task the binary classification problem of CD4+T cells and TREG cells is going to be elaborated. We aim in this classification task to identify via various classifiers to predict whether a cell, based on a variety of genes (being features/predictors), is part of the cell type/class/label “CD4+T”-cell or “TREG”-cell. Here we are using a training and test split of 80%/20% to train our classifiers on the given data of $n=5471$ (n observations, number of cells) and $p=4123$ (number of genes, predictors). After training and testing the classifiers, we are evaluating them based on given performance metrics which are accuracy, balanced accuracy, AUC and the F1 score. Ultimately, we are trying to improve the given classifiers using PCA, and finally focusing on improving the F1 score by either boosting, hyper-parameter tuning or regularisation of the classifiers. Thus, we are yielding in the end a classifier as our `mypredict()` function which is going to be used for future binary classification problems of “CD4+T” and “TREG” cells.

Task 1.1 Data Description

The `task_one_df` gives us $n=5471$ cells (classified either as “CD4+T” or as “TREG”) and respectively $p=4123$ genes (features variables) which is expressing the logarithmically normalized RNA expression level for a specific gene. In total we can find 3356 CD4+T cells and 2115 TREG cells in our total dataset. With regards to our problem, we aim to be able to predict whether a cell is a “CD4+T” cell or a “TREG” cell based on the provided features. Therefore it is of interest to find the genes with the most predictive power and information to help us classify the cells. One initial approach with the EDA would be to identify the genes which have the highest absolute difference between the mean of CD4+T cells' expression level and the mean of TREG cells' expression level. This gives intuitively the genes which can separate both cell types the best due to their difference in RNA values. First we can observe how many cells have which absolute mean difference in relation.



We can see that not all genes have a high amount of information for differentiating between the two cell types CD4+T and TREG and being thus a useful predictor for classification purposes. Rather it is a lower amount of top cells which are able to influence the classification models the most. We will discuss more on this when we talk about PCA and its principal components. In the following we can observe the summary of top 3 cells with the highest absolute difference in mean values, and in addition their boxplots.



The last plot for the third largest difference in absolute mean values differs from the other since the CD4+T 3rd quarter for TNFRSF1B is still 0 and the mean is located in the 4th quarter and therefore the box plot visually cannot be generated here.

Task 1.2 Summary Table without PCA

In the following we are evaluating seven different classifiers regarding their classification performance on the given RNA expression data set. First of all, here is a brief overview of each classifier and its classification algorithm:

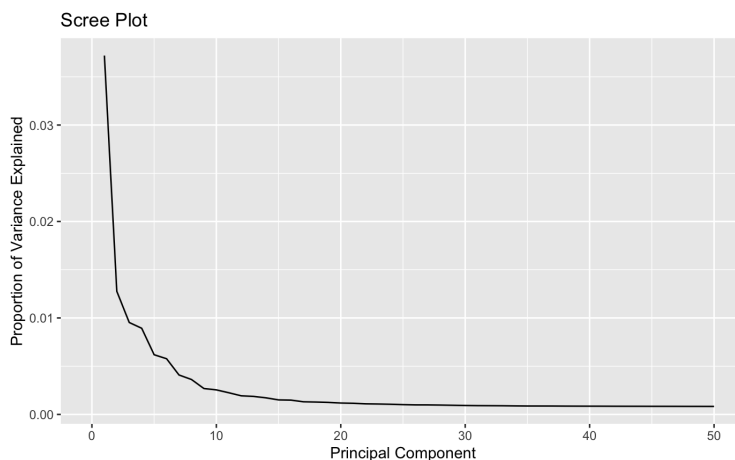
- **LDA (Linear Discriminant Analysis):** The LDA is a dimensionality reduction and uses discriminant functions (linear combinations) within the data and tries to maximize the distance of each class mean and minimize the spread of each class in order to linearly separate each classes from each other in a lower dimensional space.
- **Log Reg (Logistic Regression):** The logistic regression uses a sigmoid function to classify the cells where the genes represent the coefficients of the function. The function outputs a value between 0 and 1 and a threshold (typically 0.5) is used to either classify the cell as a CD4+T or TREG cell.
- **QDA (Quadratic Discriminant Analysis):** The QDA assumes that each class follows a Gaussian distribution with its own covariance matrix. It works by finding a decision boundary that is quadratic (curved) rather than linear, unlike Linear Discriminant Analysis (LDA). QDA models the class-specific distributions and classifies data by comparing the likelihoods of the data points belonging to each class.
- **k-NN (k-nearest Neighbor):** k-NN is a nonparametric classification algorithm which classifies a data point based on the majority class of its 'k' closest neighbors in the feature space, using a distance metric like Euclidean distance. K-NN makes predictions by examining the local structure of the data, with no explicit training phase, making it highly sensitive to the choice of 'k' and the scale of the features.
- **GBDT (Gradient Boosted Decision Trees):** Gradient Boosted Decision Trees (GBDT) are an ensemble learning method that builds a series of decision trees sequentially, where each tree corrects the errors of the previous one. It minimizes a loss function by fitting new trees to the residual errors of the combined model, improving prediction accuracy.
- **RF (Random Forest):** The RF uses bootstrapped data sets (random sampling with replacement) and uses this data for decision trees where each node is representing a subset of genes to find the best split. Moreover, multiple decision trees of different structures are made and ultimately aggregated where the majority vote classifies the cell. This makes RF already a bagged (bootstrap aggregated) classifier, allowing to reduce overfitting and improving robustness.

- **SVM (Support Vector Machine):** The SVM works by finding a hyperplane that best separates data into different classes, maximizing the margin between the classes. SVM can also handle non-linear data by using kernel functions to transform the data into higher-dimensional spaces.

w/o PCA	Accuracy	Bal. Accuracy	AUC	F1
LDA	0.9416	0.9407	0.9407	0.9238
Logistic	0.6393	0.6413	0.6597	0.5766
QDA	N/A	N/A	N/A	N/A
k-NN	0.7425	0.6964	0.6964	0.5983
GDBT	0.9087	0.8887	0.9731	0.8698
RF	0.9352	0.9171	0.9171	0.9077
SVM	0.9553	0.9479	0.9479	0.9394

Task 1.2 Summary Table with PCA

In the following we are applying the dimensionality reduction tool PCA (Principal Component Analysis). It transforms data into a new coordinate system, where the axes (principal components) capture the maximum variance in the data. It finds a set of orthogonal directions (principal components) that explain the most variability, with the first component explaining the most, the second the second most, and so on. PCA is often used to reduce the number of features while preserving as much information as possible, simplifying data for analysis or visualization. In the following scree plot we can additionally see that approximately ten principal components form the elbow of the scree plot, meaning that ten principal components account for the most Variance within the data set which makes them powerful to use for prediction models.



with PCA	Accuracy	Bal. Accuracy	AUC	F1
LDA	0.9352	0.9176	0.9892	0.9079
Logistic	0.9489	0.9414	0.9913	0.9309
QDA	0.9416	0.9298	0.9876	0.9194
k-NN	0.9187	0.8982	0.8982	0.8834
GDBT	0.8703	0.8394	0.9483	0.8060
RF	0.9379	0.9259	0.9259	0.9144
SVM (Linear Kernel)	0.9516	0.9431	0.9431	0.9342
SVM (Radial Kernel)	0.9498	0.9397	0.9397	0.9312

Task 1.3 Improved Classifiers

Now we want to focus on improving a selection of classifiers with regards to the F1 score. For this exercise we choose to improve a linear, a logistic and one more complex classifier. Hence, we are improving the LDA,

Logistic Regression and Random Forest with given improvement techniques.

	F1 Score before	F1 Score after
LDA (with CV and Hyperparameter tuning)	0.9079	0.9487
Logistic (with CV and Regularization)		
RF (with AdaBoost)	0.9141	0.9629

- **LDA with CV:** We enhance LDA by applying PCA and cross-validation (CV) to address a drop in accuracy, balanced accuracy, and F1 score when using PCA. This is likely due to fewer components/features, which reduce the model's ability to linearly separate data in lower dimensions. Cross-validation helps select the best subset for training, identifying the features that provide the most valuable information for the model. Additionally, the number of principal components and the threshold for classifying positive vs. negative predictions are tuned. Lower thresholds (around 0.2 to 0.3) improve the F1 score. The slight imbalance in the classes within the training data set is considered during hyperparameter tuning to optimize model performance.
- **Logistic CV and Lasso:** Cross-validation splits the data into multiple subsets (folds), training the model on some and testing on the others. This process helps assess the performance of logistic regression more robustly by reducing overfitting and providing a better estimate of generalization to unseen data. Regularization adds a penalty to the loss function to control model complexity, preventing overfitting by penalizing large coefficients. Lasso regularization can also perform feature selection by driving some coefficients to zero, while Ridge shrinks all coefficients towards zero without eliminating any. Together, cross-validation and regularization help evaluate and improve the model's performance, with regularization enhancing generalization. Additionally, Lasso and Ridge are compared using different values of lambda (regularization strength), while hyperparameter tuning of the logistic threshold (around 0.4) optimizes the F1 score. The number of principal components is also tuned to achieve the best F1 score.
- **RF with AdaBoost:** We choose to improve the Random Forest with AdaBoost due to the fact that the Random Forest itself is already a bagged classifier, since the Decision Trees are constructed on bootstrapped (with replacement) data sets and due to majority voting a decision is made which resembles the aggregation part of bagging. Boosting helps here to improve the weak classifiers which in this case are the Random Forests which misclassified the cells. By increasing their weights for the classification problem the classifier is focusing more on improving the false classifiers, resulting in a better performance of the Random Forest.

Task 1.4 Final predict function

Ultimately, after comparing the F1 Scores, we are choosing XXX to implement into our `mypredict()` function for Task 1.

	Accuracy	Bal. Accuracy	AUC	F1 Score
<code>mypredict()</code>				