Two-Class Classification Task[1]

Yun Yan (yy1533@nyu.edu)

Oct 20, 2015

[1] CS-GY-6923
Course Instructor: Gang Li;
TA: Xu Zhao

Given a dataset (`train.csv`) with binary labels (`train_label.csv`), this machine learning driven project is to fulfill the two-class classification task. Here in addition to data exploration, I come up with not only a comprehensive applications of models and algorithm discussed in class, but more importantly build an model using **ensemble methods**. Several pre-processing steps are involved to cook appropriate dataset before recklessly invoking standard machine learning toolbox. ROC/AUC metric is used to evaluate model performance. It shows that the blending model (with AUC = 0.990 over validating dataset) is superior to any other single model. It generates the labels of evaluation dataset (`test.csv`) as final result (`test_label.mat`).

Contents

1 Introduction

The entire workflow is shown as Figure-1.

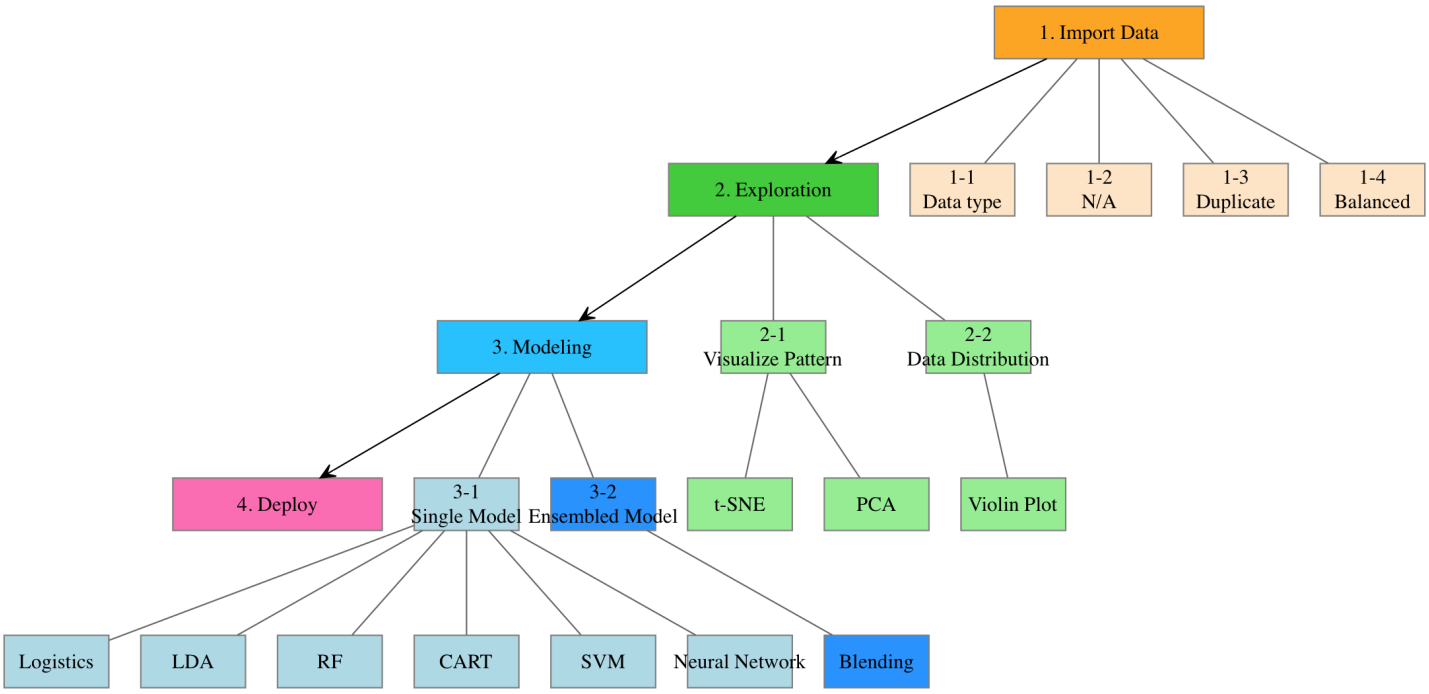

Figure 1: Workflow to solve the two-class classification problem. The workflow is mainly composed of four layers (with different color indicated).

2 Investigate dataset

Keep in mind of various data type, e.g. categorical, numerical, etc. Only after having investigated the properties of dataset can I select suitable algorithms to deal with dataset of complexity. For example, linear regression requires numerical predictors and cannot accept nominal data.

The following basics of our dataset are listed below: (i) Feature No. 43 44 45 46 47 48 49 are binary indicators, i.e. 0/1, while the rest are numbers (either integer or floating numbers). (ii) No N/A (missing value) exists. (iii) Dataset contains duplicate entries. (iv) Dataset is not balanced.

2.1 Dataset contains duplicate samples

Dataset (predictor matrix X together with one column of labels Y) $D \in \mathbb{R}^{50000\times 78}$. The number of unique entries is 48948. Duplication will interfere with model training, thus the redundant are removed and the data matrix changes down to $\mathbb{R}^{48948\times 78}$.

2.2 *Labels of samples are imbalanced*

As shown in Figure-2, the labels (also called response) of training data are highly **imbalanced**. If modeling is using imbalanced data, classification models would be trained to be biased towards one category which is in particular true for algorithms like kMeans, kNN, therefore models would be more likely to fail on classifying unseen data.

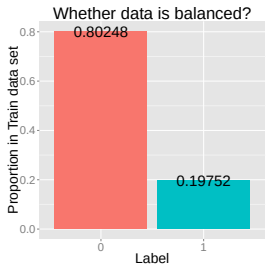

Figure 2: Barplot indicating the proportion of each label within the training dataset. The percentage of Label-0 and Label-1 is roughly 80% and 20% respectively.

2.3 *Data is potentially separable*

Long before training classifier, I was wondering whether the data is separable. If intrinsic properties of data itself implied there is no way to be classified, more work on algorithms are hopeless. Therefore I would like to use **un-supervised models** to investigate and visualize the underlying pattern.

PCA was introduced during EigenFace class and considered as promising dimension reduction method, hence here PCA is applied on normalized predictor matrix X and the first 2 PCs are extracted to visualize the entire predictor in 2-D scatter plot (See Figure-3).

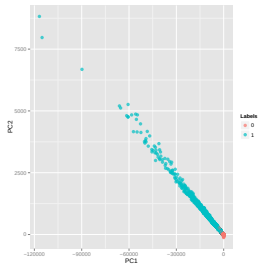

Figure 3: Scatterplot showing the underlying structure of predictor matrix with ranking top-2 principle components as axises. Afterwards the color of each entry indicating its label is added.

However, the PCA scatter plot could hardly reveal data pattern. This result is plausible because the first two PCs poorly explain variance thus much information are lost (See Figure-4) and cannot be proper representatives of original full data.

I turn to t-SNE which is an un-supervised feature reduction approach and quite popular among Kaggle competition.

Figure-5 intuitively indicates the underlying structure of data is potentially separable within 2D space thus it gives me hope to figure out solutions for the two-class classification task on higher dimensional space.

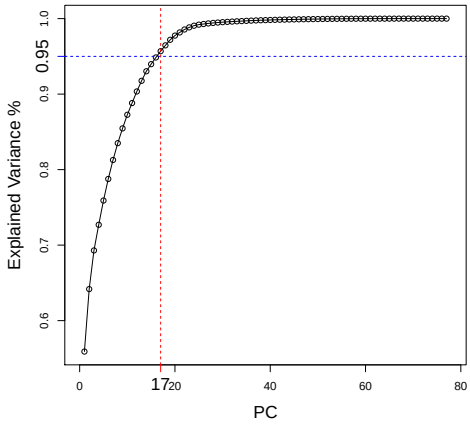

Figure 4: Line plot showing the cumulatively explained variance over ordering principle components. Top 17 PCs (indicated by red dashed line) could explain at least 95% variance (indicated by blue dashed line). Top 2 PCs can only explain 60-65%.

3 *Prepare training dataset and validating dataset*

`train.csv` (i.e. predictor matrix X) and `train_label.csv` (i.e. response vector Y) together form our dataset D. Unique entries dataset U is produced from raw dataset D with duplication removed.

As one of the strategies to avoid over-fitting, dataset U will be randomly split into two subset dataset, i.e. training dataset T and validating dataset V, which takes **80% and 20%** respectively. The dataset T is for training model and dataset V is for testing model performance.

Therefore validating dataset V here is actual test dataset. Because`test.csv` is supposed for evaluation instead of testing but it uses the name already, here comes word "validating" for dataset V to maintain naming consistency.

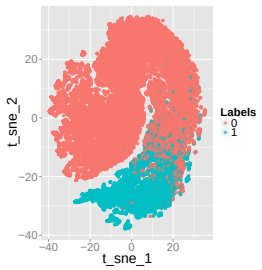

Figure 5: Scatterplot showing the result of t-SNE on unique predictor data matrix. Color indicating label of each entry is added afterwards.

4 Modeling and tuning

Having tested logistic regression with different training settings (See Appendix-A Figure-12), I decide to perform **5-repeated 10-fold cross validation**.

Furthermore to address data unbalance issue, subsampling method is used to recover balance of training dataset. For example, a classic method, down-sampling, will down-sample entries with Label-0 to same size as Label-1. But here a more complexed over-sampling approach is applied, which is called "**Random Over-Sampling Examples**" implemented in `ROSE` R language package (See Methods).

In sum, I decide to perform **subsampling during cross validation** to train models.

Because it is cross validation by which fashion the models are learning training dataset, ROC/AUC is available as metric for convergence condition. More importantly AUC is favored over accuracy, in particular our dataset is imbalanced.

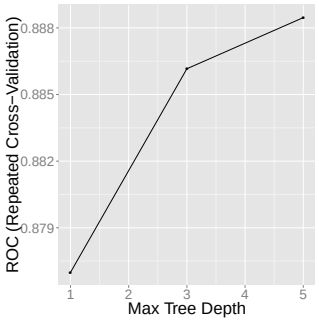

Figure 6: Tuning CART model with 3 settings

4.1 Logistic Regression

None

4.2 LDA

None

4.3 CART

Max Tree Depth: Greater the depth, more-likely overfitting. See Figure-6.

The trained tree structure is shown as Figure-7.

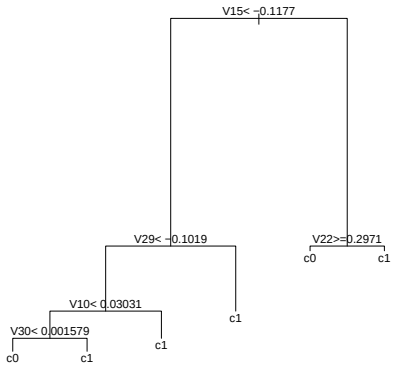

Figure 7: Tree structure trained by CART algorithm.

4.4 Random Forest

Number of randomly selected predictors: Less predictors are selected, more-likely under-fitting. I tried three sets by selecting 5, 25, 55 predictors. See Figure-8.

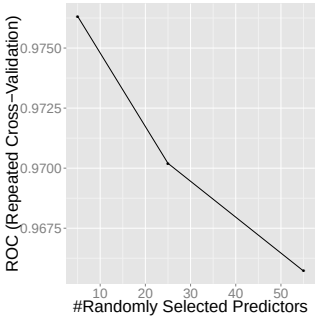

Figure 8: Tuning RF model with 3 settings

4.5 SVM (Linear)

Cost: Constant of the regularization term in the Lagrange formulation to control the soft margin. I set it as constant $C = 1$ thus no tuning for linear SVM.

4.6 *Neural Network*

1) *Number of hidden units*: more hidden units, more-likely overfitting;
2) *Weight decay*: controls the training process.
   See Figure-9.

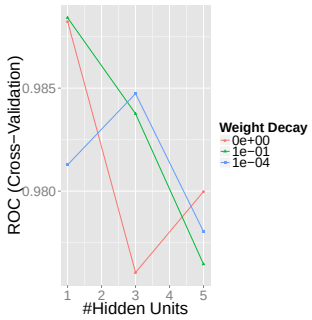

Figure 9: Tuning neural network with 3 settings

4.7 *Ensemble Model*

Having trained all the above 7 models, I turn to an advanced topic : ensemble model which combines the power of wisdom of group [2]. Specifically **blending** method fits my need.

I set up a "voting committee" with all the 7 trained models as members to determine the label of unseen data. The committee is implemented by logistic regression model for convenience, and each model trained previously is one feature of the model.

In details, each model will report the probability of unseen sample being "Label-0", rather than categorical label. The results of individual models will be feature spaces of ensemble model which is a new logistic regression model. The work for constructing the blending model is equivalent to building a logistic regression with numerical inputs and binary labels as output.

[2] MLWave. Kaggle ensembling guide, June 2015. URL `http://mlwave.com/kaggle-ensembling-guide/`

5 *Performance and Blending model is best*

Performance of all 7 models is shown as Figure-10. Logistic regression and SVM (Linear kernel) are top best individual models.

Shown as Figure-11, blending model (AUC = 0.990) is superior to any other individual model, except logistic regression and SVM. Blending model is neither dominated by nor exactly same as individual models (results not shown here), therefore having same AUC values does not imply they are same model, but rather means they have same performance and prediction ability

Furthermore, the unseen test (evaluation) dataset is expected to be complexed, therefore it is better to deploy blending model which take multiple factors into account.

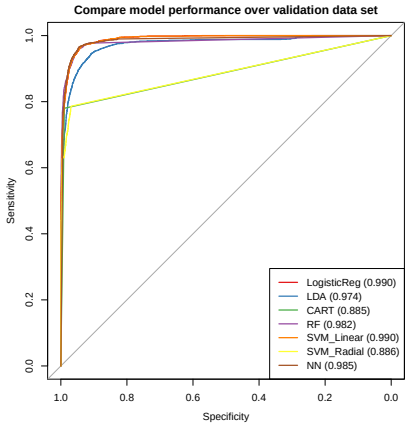

Figure 10: Performance of 7 models, i.e. logistics regression, LDA, RF, CART, SVM(linear), SVM(radial), Neural Network. ROC/AUC is estimated by applying models on independent validating dataset and AUC values are attached to model names on bottom right.

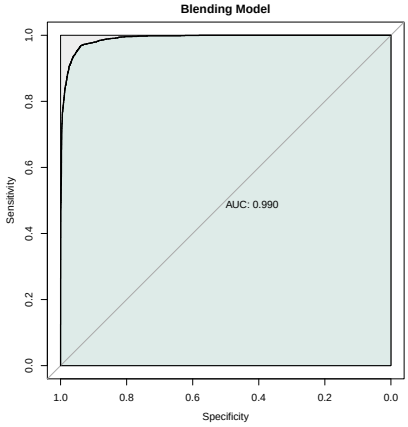

Figure 11: Performance of blending model. ROC/AUC is estimated by applying model on independent validating dataset. Blending model achieves AUC = 0.990, which is superior to any other individual model.

6 *Deploy*

Read-in the `test.csv` and feed its predictor matrix to my blending model, so that labels are generated as final results.

7 Methods

Entire workflow is implemented by using R language with standard packages, in particular `caret`[3] (short for Classification And REgression Training). It attempts to provides uniform syntax to call other existed R packages to build machine learning workflow.

Compared with Matlab requiring commercial licenses, R is a free software environment for statistical computing and graphics. More important, it can natively employ C/C++ codes to implement packages to avoid running slow, while Matlab is mostly slow. As our data is relatively big and advanced algorithm are involved, e.g. SVM, random forest, neural network, it is wise to select a language with both professional statistics and machine learning ability and also running efficiency.

Entire work was designed and implemented independently by Yun Yan.

[3] Max Kuhn. The caret package, Nov 2015. URL `http://topepo.github.io/caret/index.html`

7.1 t-SNE: unique predictor v.s. unique sample

Unique predictor matrix and unique samples are two different concepts because sample = predictor vector + label scalar. Two samples might share the same predictor vector but differ at binary label. It is important to investigate these possible events, because in this case model learning becomes much more challenging. In addition, only with extra data pre-processing, can t-SNE [4] be used to intuitively check whether dataset is separable. As unsupervised learning technique, t-SNE accepts predictor matrix alone. It is afterwards when label information is added into 2D scatterplot. There is labeling collision for two samples with same predictor vector but distinct labels.

Good news is that these cases do not exist in our dataset, which allows me to directly apply t-SNE for data visualization.

[4] Laurens van der Maaten. t-sne, 2008. URL `https://lvdmaaten.github.io/tsne/`; and Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(2579-2605): 85, 2008

7.2 Model training packages

Model	**Method of `caret`**	**R package**
Logistic Regression	`glm`	stats
LDA	`lda`	MASS
CART	`rpart2`	rpart
Random Forest	`rf`	randomForest
SVM (Linear)	`svmLin`	kernlab
Neural Network	`nnet`	nnet

Table 1: List of dependencies for implementation. Second column gives the `method` name required by `caret` package to perform selected algorithm.

7.3 *General usage of `caret`*

First is the general settings to perform subsampling (ROSE) during 5-times repeated 10-fold cross validation.

```
trCtrolRepCV <- trainControl(method = 'repeatedcv', number = 10,
                             repeats = 5,
                             sampling = 'rose',
                             classProbs = TRUE,
                             summaryFunction = twoClassSummary)
```

Then we take LDA as example. It is by calling function named `train` to run model learning. Before actual modeling, `train` function will normalize dataset as required by user. Finally "ROC" is used as `metric`.

```
mdl_lda <- train(Label ~ . ,       # Formula: Label ~ Response matrix
                 data = train_df, # Dataset D = [X | Y]
                 method = "lda",
                 trControl = trCtrolRepCV,
                 preProcess = c('center', 'scale'),
                 metric = "ROC")
```

The other individual models share the similar syntax.

7.4 *Platform and software versions*

Ubuntu 14.04.3 LTS (GNU/Linux 3.13.0-68-generic x86_64)
Intel(R) Xeon(R) CPU E5-2630L v2 @ 2.40GHz
R version 3.2.2 (2015-08-14)

8 *Appendix-A*

In order to determine general training strategy, logistic regression model is trained to learn demo training dataset (random 1500 samples).

8.1 *Highlight usage of repeated CV*

Using logistic regression as basic model, I tested performance with different CV settings: repeated CV v.s. CV alone (See Figure-12). Hence 5-times repeated 10-fold CV is superior to 10-fold CV alone, admittedly AUC did not improve much.

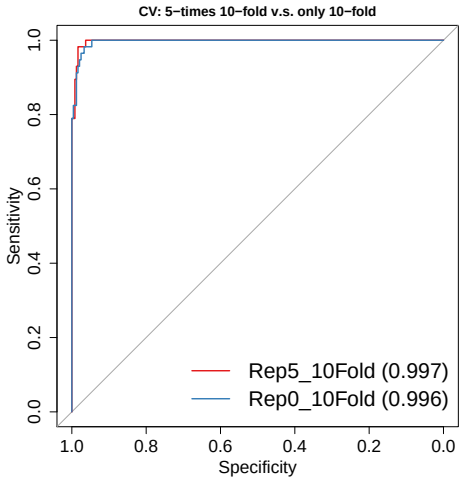

Figure 12: ROC curve for logistic regression model with different training settings: 10-fold CV alone v.s. 5-repeated 10-fold CV. AUC is calculated by applying model on independent validating dataset.

8.2 *Highlight usage of subsampling*

To deal with imbalanced training dataset, I investigated whether subsampling during resampling improves model performance. The

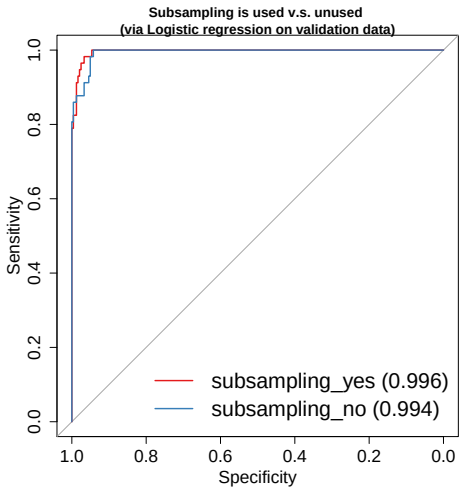

Figure 13: ROC curve for logistic regression model with different training settings: Subsampling during CV v.s. Without subsampling. AUC is calculated by applying model on independent validating dataset.

answer is yes, as expected (See Figure-13).

9 *Appendix-B: Matlab*

This appendix shows how I train 4 models with KNN (with/without PCA), Linear Discriminant, SVM algorithms by using **Matlab** alone, specifically Statistics and Machine Learning Toolbox which has friendly GUI.

Firstly, when importing data via graphic interface (See Figure-14). `train.csv` and `train_label.csv` are merged before being imported. Specify the last column (i.e. label) of merged dataset as "response" in the GUI.

Secondly, train models. On top-left "Data browser" window, I have already trained 4 models, marked with accuracy on train dataset. The first one is KNN with cosine similarity as distance metric (See Figure-15). It turns on PCA option to do dimension reduction. The detailed specs are available on bottom-left window. On right panel, its performance over training dataset is available, e.g. its AUC is 0.97. Noticed that these information about ROC curve can NOT be used to evaluate model performance because AUC is estimated over training set, rather than test test.

Third, it is straightforward to train other models with click-run operations thanks to GUI. See the results as shown in Figure-15, 16, 17, 18.

Finally deploy and predict. Save the models into Matlab workspace and predict labels for test (evaluation) dataset. Results are not shown here because these are not final submitted results.

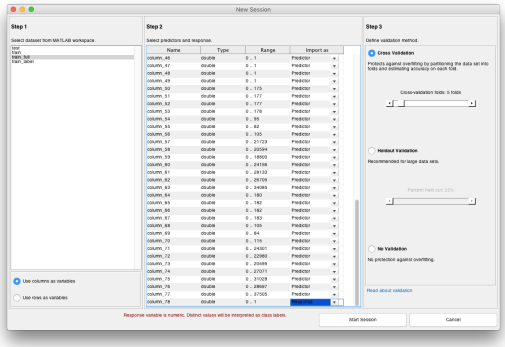

Figure 14: Importing data from Matlab workspace with statistic and machine learning toolbox.

References

Max Kuhn. The caret package, Nov 2015. URL `http://topepo.github.io/caret/index.html`.

MLWave. Kaggle ensembling guide, June 2015. URL `http://mlwave.com/kaggle-ensembling-guide/`.

Laurens van der Maaten. t-sne, 2008. URL `https://lvdmaaten.github.io/tsne/`.

Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(2579-2605):85, 2008.

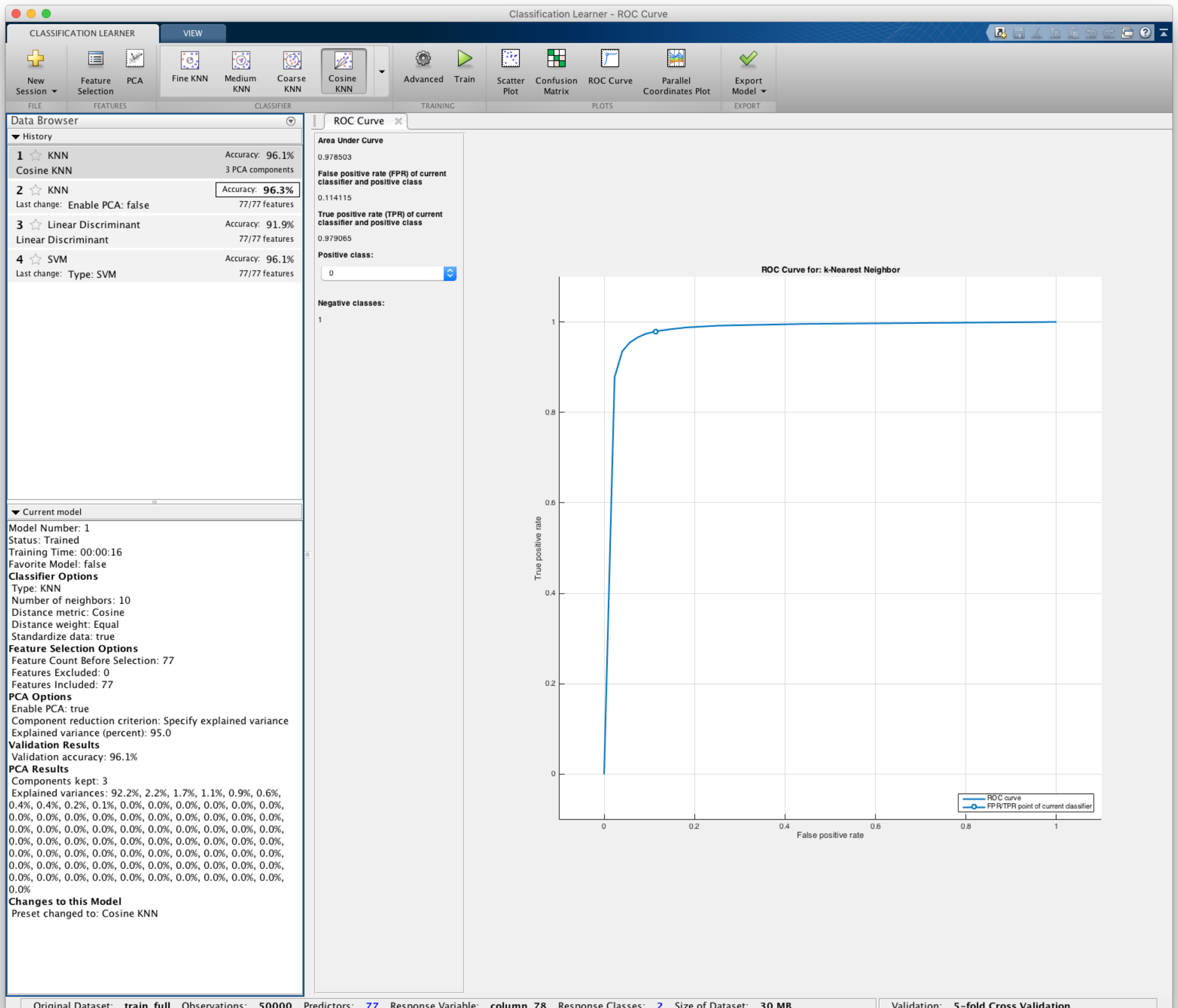

Figure 15: First KNN model details. Distance metric: cosine similarity; PCA: enabled.

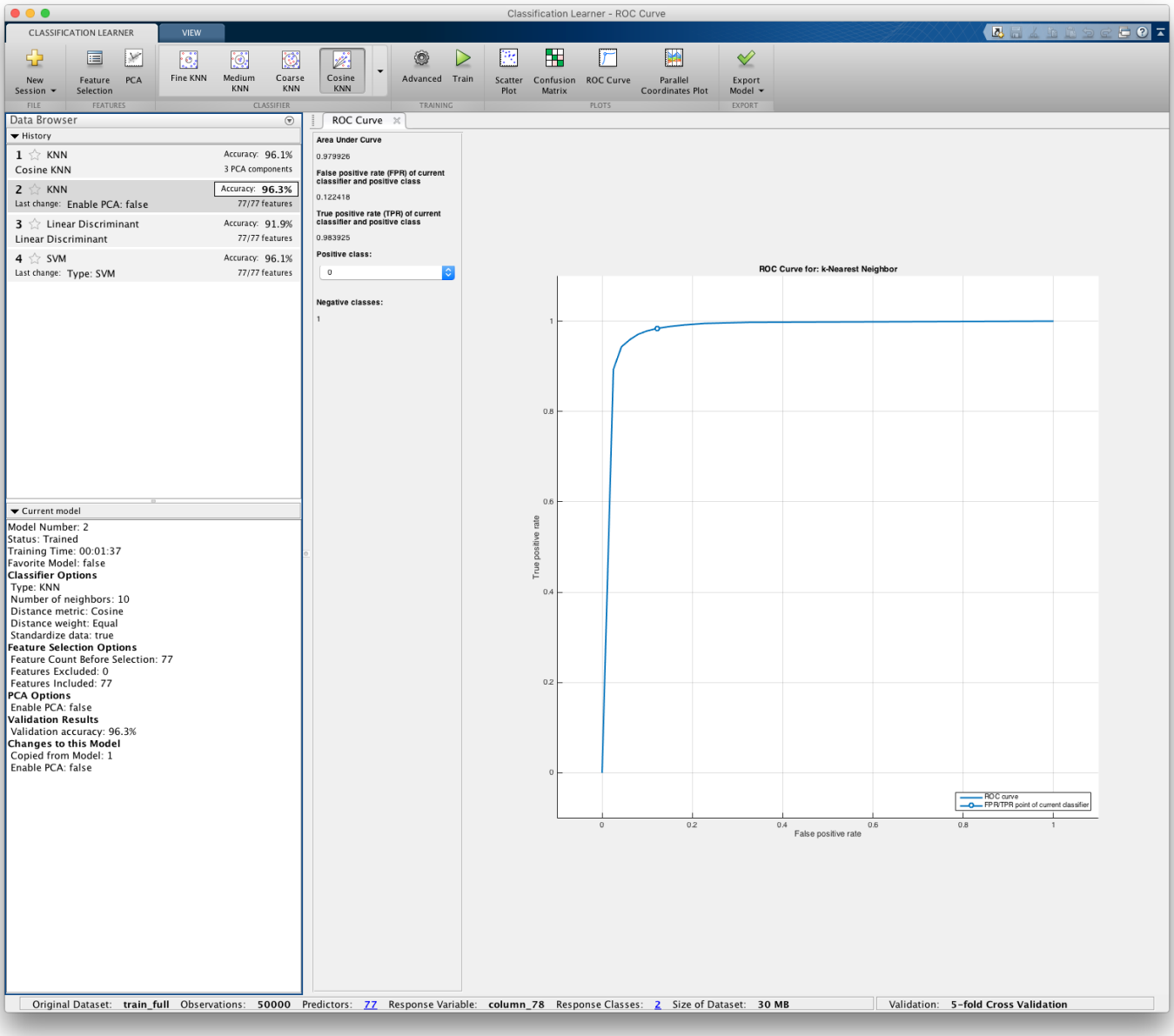

Figure 16: Second KNN model details.Distance metric: cosine similarity; PCA: disabled.

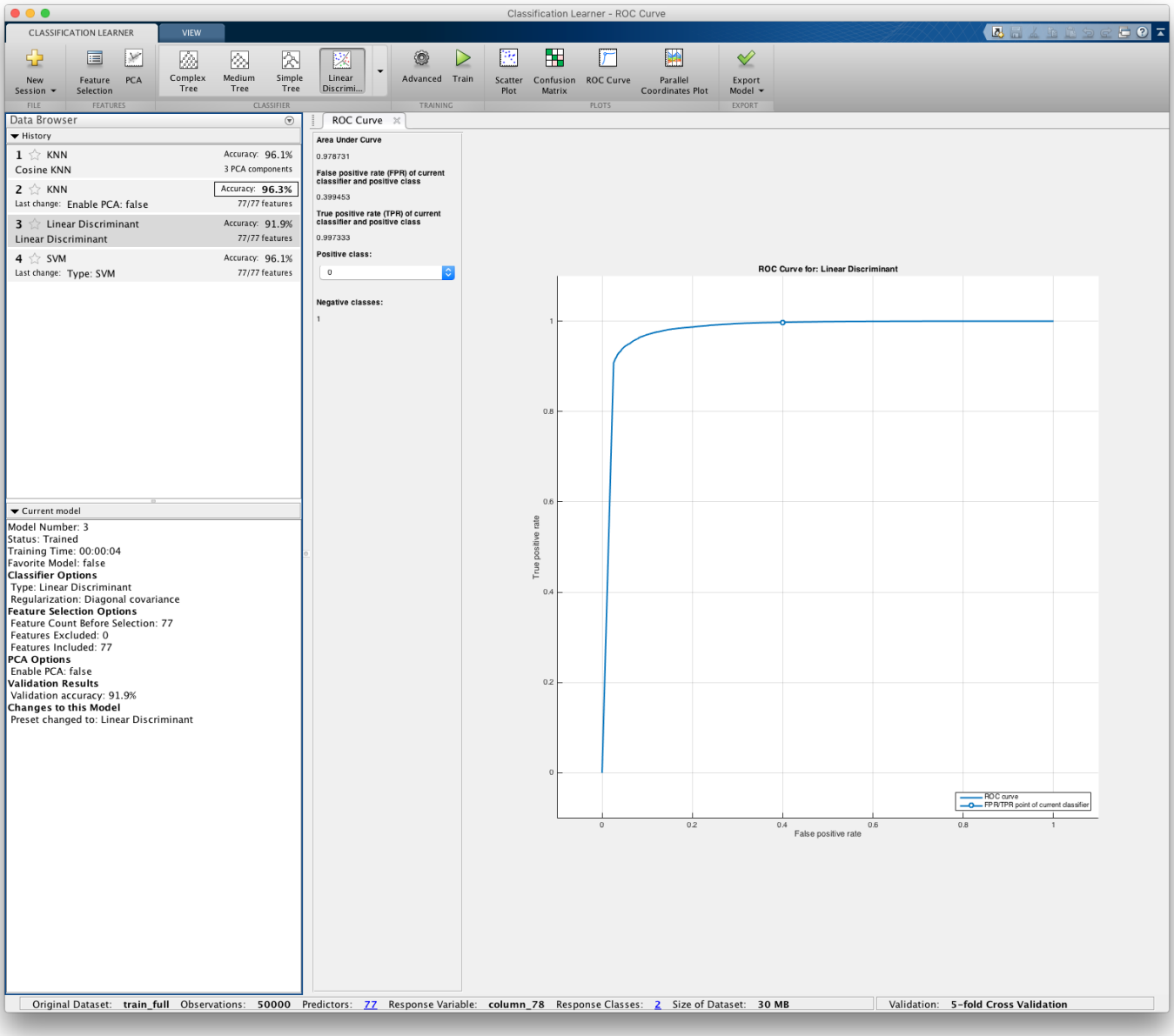

Figure 17: Third model: LDA.

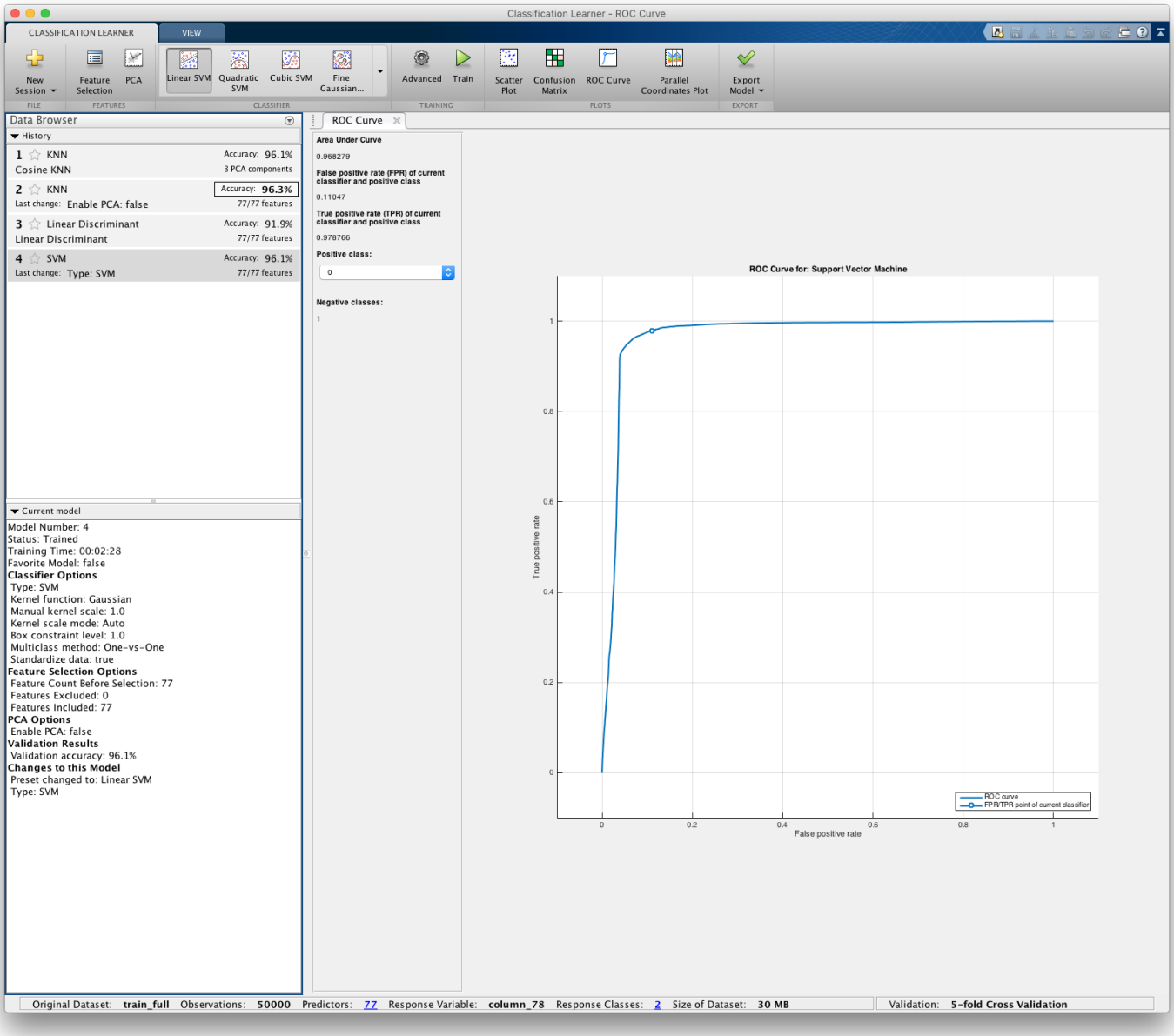

Figure 18: Fourth model: SVM (Gaussian kernel).