# Ovarian Cancer Histotypes: Report of Statistical Findings

Derek Chiu

2024-03-25

# Contents

# List of Figures

# List of Tables

# Preface

This report of statistical findings describes the classification of ovarian cancer histotypes using data from NanoString CodeSets.

Marina Pavanello conducted the initial exploratory data analysis, Cathy Tang implemented class imbalance techniques, Derek Chiu conducted the normalization and statistical analysis, and Lauren Tindale and Aline Talhouk are the project leads.

# 1. Introduction

Ovarian cancer has five major histotypes: high-grade serous carcinoma (HGSC), low-grade serous carcinoma (LGSC), endometrioid carcinoma (ENOC), mucinous carcinoma (MUC), and clear cell carcinoma (CCOC). A common problem with classifying these histotypes is that there is a class imbalance issue. HGSC dominates the distribution, commonly accounting for 70% of cases in many patient cohorts, while the other four histotypes are spread over the rest of the cases. Subsampling methods like up-sampling, down-sampling, and SMOTE can be used to mitigate this problem.

The supervised learning is performed under a consensus framework: we consider various classification algorithms and use evaluation metrics like accuracy, F1-score, Kappa, and G-mean to inform the decision of which methods to carry forward for prediction in confirmation and validation sets.

# 2. Methods

## 2.1 Normalization

The full training set was comprised of data from CodeSet (CS) 1, 2, and 3. All CodeSets were first normalized to housekeeping genes, then a different approach was taken for each of the CodeSets.

CS1 was normalized to CS3 using "Random1" reference samples. These reference samples are common samples between CS1 and CS3, randomly selected such that we obtain one from each of the five histotypes. Then we use the reference method to normalize CS1 to CS3.

Similarly, CS2 was normalized to CS3 using "Random1" reference samples using five common samples between CS2 and CS3 such that there is one from each histotype.

For CS3, we first split the dataset by site: Vancouver, USC, and AOC. We use the CS3-Vancouver subset as a "reference standard", so we normalized CS3-USC and CS3-AOC to CS3-Vancouver using a "Random1" reference method where we reference samples are common between USC and Vancouver, and between AOC and Vancouver. The CS3-Vancouver is also included without further normalization.

## 2.2 Case Selection

Duplicate cases (two samples with the same ottaID) were removed from the training set before fitting the classification models. CS3 cases were preferred over CS1 and CS2, and CS3-Vancouver were preferred over CS3-AOC and CS3-USC.

The training, confirmation, and validation sets all used a different set of cohorts.

## 2.3 Classifiers

We use 4 classification algorithms in the supervised learning framework for the Training Set. The pipeline was run using SLURM batch jobs submitted to a partition on a CentOS 7 server. All resampling techniques, pre-processing, model specification, hyperparameter tuning, and evaluation metrics were implemented using the `tidymodels` suite of packages. The classifiers we used are:

- Random Forest (`rf`)
- Support Vector Machine (`svm`)
- XGBoost (`xgb`)
- Regularized Multinomial Regression (`mr`)

### 2.3.1 Resampling of Training Set

We used a nested cross-validation design to assess each classifier while also performing hyperparameter tuning. An outer 5-fold CV stratified by histotype was used together with an inner 5-fold CV with 2 repeats stratified by histotype. This design was chosen such that the test sets of the inner resamples would still have a reasonable number of samples belonging to the smallest minority class.

### 2.3.2 Hyperparameter Tuning

The following specifications for each classifier were used for tuning hyperparameters:

- `rf` and `xgb`: The number of trees were fixed at 500. Other hyperparameters were tuned across 10 randomly selected points in a latin hypercube design.
- `svm`: Both the cost and sigma hyperparameters were tuned across 10 randomly selected points in a latin hypercube design within ranges (transformed scale) [0, 2] and [-3, 0], respectively.
- `mr`: We generated 10 randomly selected points in a latin hypercube design for the penalty (lambda) parameter. Then, we generated 10 evenly spaced points in [0, 1] for the mixture (alpha) parameter in the regularized multinomial regression model. These two sets of 10 points were crossed to generate a tuning grid of 100 points.

### 2.3.3 Subsampling

Here are the specifications of the subsampling methods used to handle class imbalance:

- None: No subsampling is performed
- Down-sampling: All levels except the minority class are sampled down to the same frequency as the minority class
- Up-sampling: All levels except the majority class are sampled up to the same frequency as the majority class
- SMOTE: All levels except the majority class have synthetic data generated until they have the same frequency as the majority class
- Hybrid: All levels except the majority class have synthetic data generated up to 50% of the frequency of the majority class, then the majority class is sampled down to the same frequency as the rest.

The figure below helps visualize how the distribution of classes changes when we apply subsampling techniques to handle class imbalance:

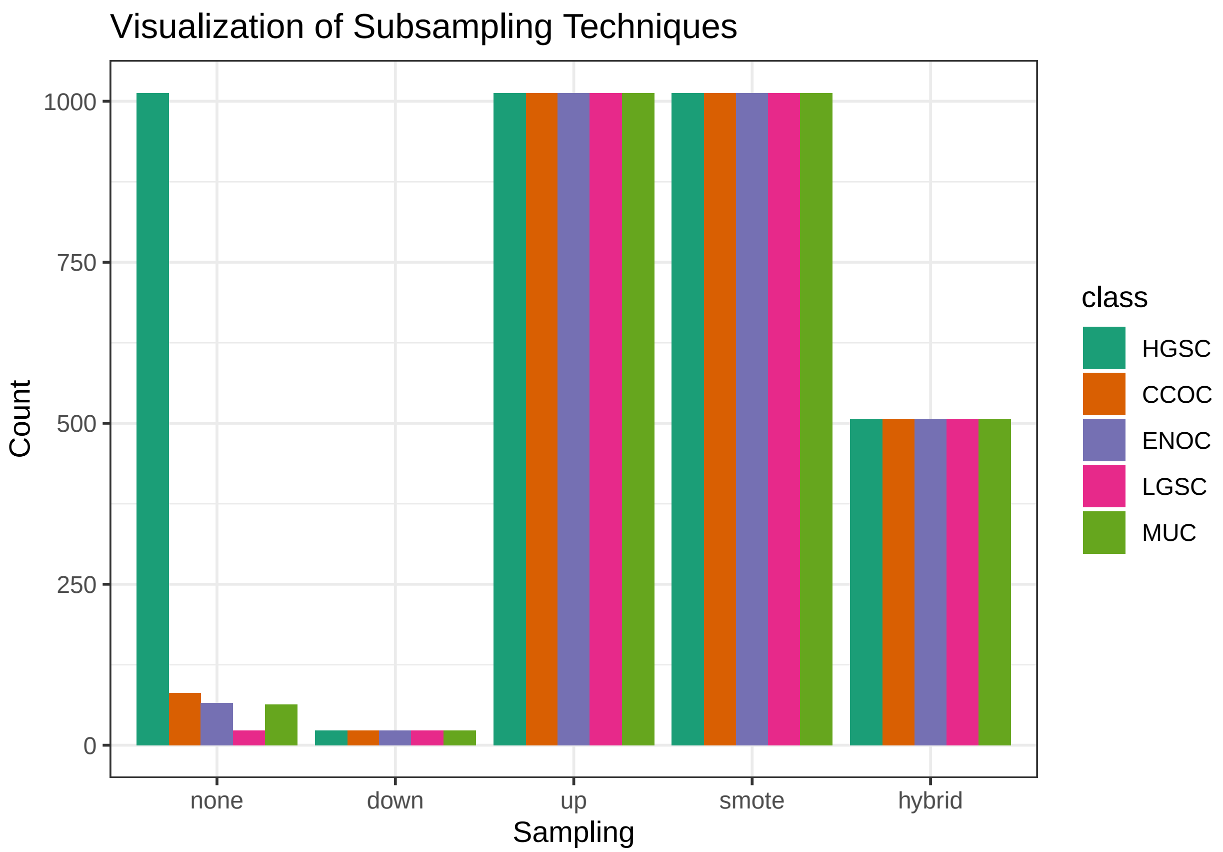


Figure 2.1: Visualization of Subsampling Techniques

## 2.4 Sequential Algorithm

Instead of training on k classes simultaneously using multinomial classifiers, we can use a sequential algorithm that performs k-1 one-vs-all binary classifications iteratively to obtain a final prediction of all cases. At each step in the sequence, we classify one class vs. all other classes, where the classes that make up the “other” class are those not equal to the current “one” class and excluding all “one” classes from previous steps. For example, if the “one” class in step 1 was HGSC, the “other” classes would include CCOC, ENOC, LGSC, and MUC. If the “one” class in step 2 was CCOC, the “other” classes include ENOC, LGSC, and MUC.

The order of classes and workflows to use at each step in the sequential algorithm must be determined using a retraining procedure. After removing the data associated with a particular class, we retrain using the remaining data using multinomial classifiers as described before. The class and workflow to use for the next step in the sequence is selected based on the best per-class evaluation metric value (e.g. F1-score).

Let

$$\begin{aligned} X_k &= \text{Training data with k classes} \\ C_k &= \text{Class with highest } F_1 \text{ score from training } X_k \\ W_k &= \text{Workflow associated with } C_k \end{aligned}$$

Figure 2.2 illustrates how the sequential algorithm works for K=5, using ovarian histotypes as an example for the classes.

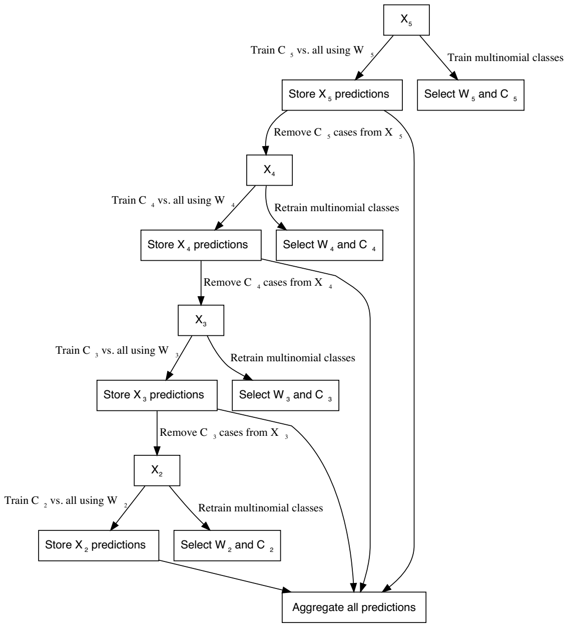

$X_5$
Train $C_5$ vs. all using $W_5$
Train multinomial classes
Store $X_5$ predictions
Select $W_5$ and $C_5$
Remove $C_5$ cases from $X_5$
$X_4$
Train $C_4$ vs. all using $W_4$
Retrain multinomial classes
Store $X_4$ predictions
Select $W_4$ and $C_4$
Remove $C_4$ cases from $X_4$
$X_3$
Train $C_3$ vs. all using $W_3$
Retrain multinomial classes
Store $X_3$ predictions
Select $W_3$ and $C_3$
Remove $C_3$ cases from $X_3$
$X_2$
Train $C_2$ vs. all using $W_2$
Retrain multinomial classes
Store $X_2$ predictions
Select $W_2$ and $C_2$
Aggregate all predictions

Figure 2.2: Sequential Algorithm

### 2.4.1 Subsampling

The subsampling method used in the first step of the sequential algorithm is used in all subsequent steps in order to maintain data pre-processing consistency. As a result, we are only comparing classification algorithms within one subsampling method across the entire sequential algorithm.

## 2.5 Two-Step Algorithm

The HGSC histotype comprises of approximately 80% of cases among ovarian carcinoma patients, while the remaining 20% of cases are relatively, evenly distributed among ENOC, CCOC, LGSC, and MUC histotypes. We can implement a two-step algorithm as such:

- Step 1: use binary classification for HGSC vs. non-HGSC
- Step 2: use multinomial classification for the remaining non-HGSC classes

Figure 2.3 shows how the two-step algorithm works:

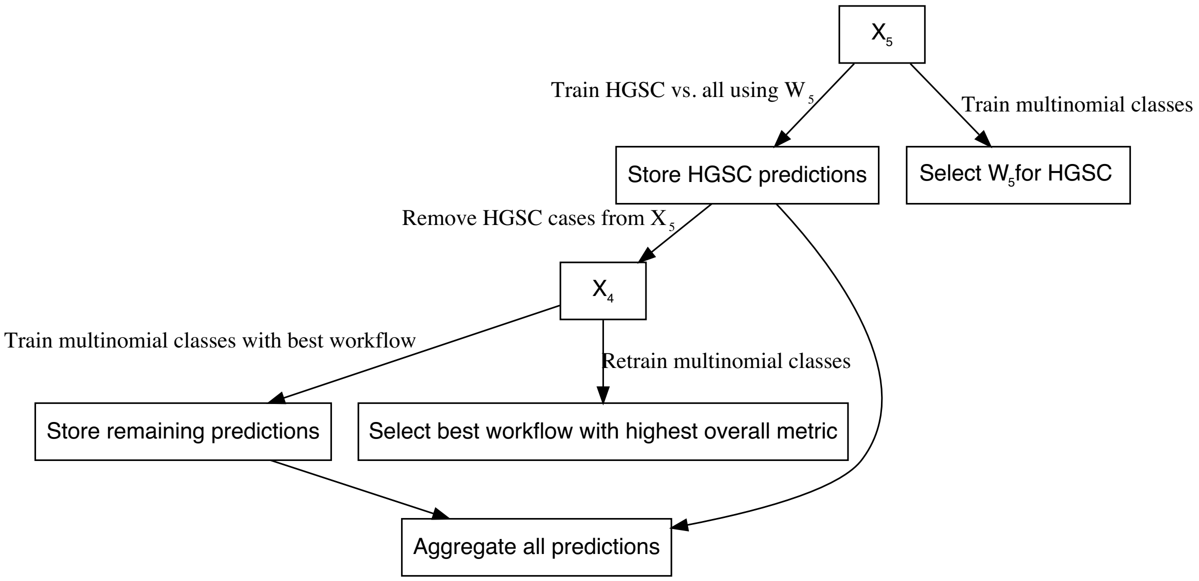

$X_5$
Train HGSC vs. all using $W_5$
Train multinomial classes
Store HGSC predictions
Select $W_5$for HGSC
Remove HGSC cases from $X_5$
$X_4$
Train multinomial classes with best workflow
Retrain multinomial classes
Store remaining predictions
Select best workflow with highest overall metric
Aggregate all predictions

Figure 2.3: Two-Step Algorithm

### 2.5.1 Subsampling

Although the class imbalance problem is mostly eliminated in Step 2 after removing the HGSC cases, we still use the same subsampling method in Step 2 as was used in Step 1 to keep the algorithm consistent.

## 2.6 Gene Optimization

We want to discover an optimal set of genes for the classifiers while including specific genes from other studies. A total of 72 genes are used in the classifier training set.

There are 16 genes in the classifier set that overlap with the PrOTYPE classifier: COL11A1, CD74, CD2, TIMP3, LUM, CYTIP, COL3A1, THBS2, TCF7L1, HMGA2, FN1, POSTN, COL1A2, COL5A2, PDZK1IP1, FBN1

There are also 13 genes in the classifier set that overlap with the SPOT signature: HIF1A, CXCL10, DUSP4, SOX17, MITF, CDKN3, BRCA2, CEACAM5, ANXA4, SERPINE1, TCF7L1, CRABP2, DNAJC9.

Taking the union of PrOTYPE and SPOT genes we obtain a total of 28 unique genes that we want to use for the final classifier, regardless of model performance. We then incrementally add genes from the remaining 44 candidate genes based on an overall variable importance rank to this list and recalculate performance metrics. The number of genes at which the performance peaks or starts to plateau may indicate an optimal gene set model for us to compare with the full set model.

### 2.6.1 Variable Importance

Variable importance is calculated using either a model-based approach if it is available, or a permutation-based VI score otherwise (e.g. for SVM). The variable importance scores are averaged across the outer training folds, and then ranked from highest to lowest.

For the sequential and two-step classifiers, we calculate an overall VI rank by taking the cumulative union of genes at each variable importance rank across all sequences, until all genes have been included.

# 3. Distributions

## 3.1 Histotypes in Classifier Data

## 3.2 Cohort Counts

## 3.3 Cohorts in Classifier Data

## 3.4 Quality Control

### 3.4.1 Failed Samples

We use an aggregated `QCFlag` that considers a sample to have failed QC if any of the following conditions are true:

- `linFlag`: linearity of positive controls with positive control concentrations is less than 0.95, or linearity measures are unknown
- `imagingFlag`: percent of field of view is less than 75%
- `spcFlag`: smallest positive control is less than the lower limit of detection (negative control average expression less two times the negative control standard deviation), or negative control average expression equals zero
- `normFlag`: signal to noise ratio less than 100, or percent of genes detected is less than 50. Note: these thresholds were determined by examining the %GD vs. SNR relationship below.

### 3.4.2 %GD vs. SNR

\begin{figure}[H]

Table 3.1: Pre-QC Training Set Histotype Distribution by CodeSet

| Variable | Levels | CS1 | CS2 | CS3 | Total |
|---|---|---|---|---|---|
| Histotype | HGSC | 120 (45%) | 643 (79%) | 515 (92%) | 1278 (78%) |
| | CCOC | 48 (18%) | 61 (7%) | 11 (2%) | 120 (7%) |
| | ENOC | 60 (22%) | 32 (4%) | 11 (2%) | 103 (6%) |
| | MUC | 19 (7%) | 62 (8%) | 12 (2%) | 93 (6%) |
| | LGSC | 20 (7%) | 21 (3%) | 9 (2%) | 50 (3%) |
| Total | N (%) | 267 (16%) | 819 (50%) | 558 (34%) | 1644 (100%) |

Table 3.2: Training Set (with duplicates) Histotype Distribution by CodeSet

| Variable | Levels | CS1 | CS2 | CS3 | Total |
|---|---|---|---|---|---|
| Histotype | HGSC | 116 (48%) | 623 (80%) | 475 (94%) | 1214 (79%) |
| | CCOC | 44 (18%) | 54 (7%) | 8 (2%) | 106 (7%) |
| | ENOC | 55 (23%) | 27 (3%) | 8 (2%) | 90 (6%) |
| | MUC | 15 (6%) | 59 (8%) | 9 (2%) | 83 (5%) |
| | LGSC | 14 (6%) | 19 (2%) | 6 (1%) | 39 (3%) |
| Total | N (%) | 244 (16%) | 782 (51%) | 506 (33%) | 1532 (100%) |

Table 3.3: Final Training Set Histotype Distribution by CodeSet

| Variable | Levels | CS1 | CS2 | CS3 | Total |
|---|---|---|---|---|---|
| Histotype | HGSC | 9 (12%) | 553 (79%) | 451 (96%) | 1013 (81%) |
| | CCOC | 25 (32%) | 52 (7%) | 4 (1%) | 81 (7%) |
| | ENOC | 37 (48%) | 25 (4%) | 4 (1%) | 66 (5%) |
| | MUC | 3 (4%) | 55 (8%) | 5 (1%) | 63 (5%) |
| | LGSC | 3 (4%) | 16 (2%) | 4 (1%) | 23 (2%) |
| Total | N (%) | 77 (6%) | 701 (56%) | 468 (38%) | 1246 (100%) |

Table 3.4: Histotype Distribution in Confirmation and Validation Sets

| Variable | Levels | Confirmation | Validation |
|---|---|---|---|
| Histotype | HGSC | 422 (66%) | 674 (74%) |
| | CCOC | 75 (12%) | 80 (9%) |
| | ENOC | 106 (16%) | 108 (12%) |
| | MUC | 27 (4%) | 26 (3%) |
| | LGSC | 13 (2%) | 18 (2%) |
| Total | N (%) | 643 (42%) | 906 (58%) |

Table 3.5: Training Set counts by CodeSet and Processing Stage

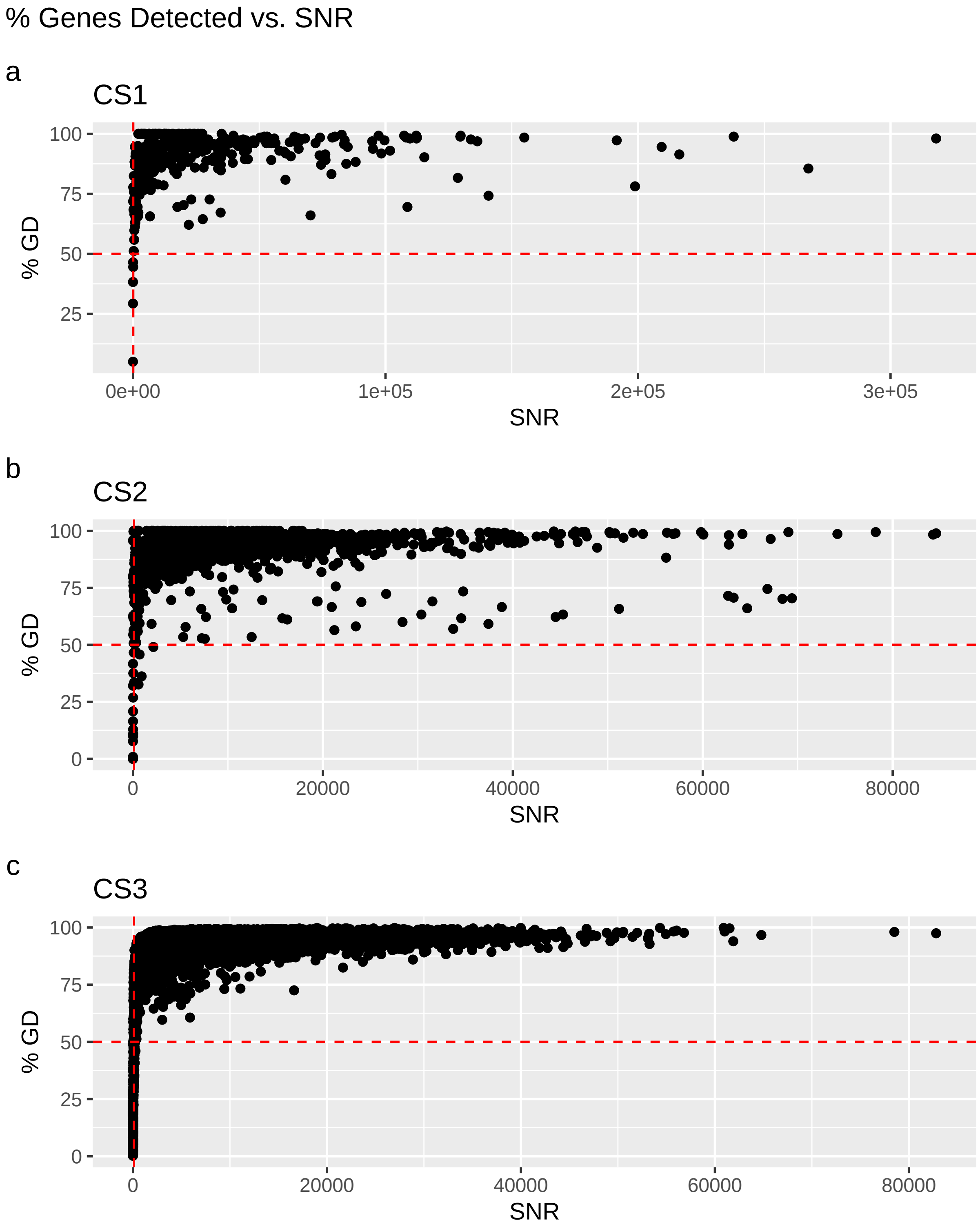
| Processing Stage | CS1 | CS2 | CS3 | Total |
|---|---:|---:|---:|---:|
| Raw Data | 412 | 1223 | 5424 | 7059 |
| Selected Cohorts | 294 | 903 | 2477 | 3674 |
| QC | 286 | 888 | 2285 | 3459 |
| Normalized to Reference | 263 | 832 | 2107 | 3202 |
| CS3: remove test sets, add AOC/USC | 263 | 832 | 514 | 1609 |
| Major Histotypes | 244 | 782 | 506 | 1532 |
| Removed Duplicates | 77 | 701 | 468 | 1246 |

Table 3.6: Cohort Distribution in Training, Confirmation, and Validation Sets

| CodeSet | Cohort | Training | Confirmation | Validation |
|---|---|---|---|---|
| CS1 | MAYO | 2 | 0 | 0 |
| CS1 | MTL | 1 | 0 | 0 |
| CS1 | OOU | 53 | 0 | 0 |
| CS1 | OOUE | 1 | 0 | 0 |
| CS1 | VOA | 20 | 0 | 0 |
| CS2 | ICON7 | 365 | 0 | 0 |
| CS2 | JAPAN | 8 | 0 | 0 |
| CS2 | MAYO | 42 | 0 | 0 |
| CS2 | MTL | 59 | 0 | 0 |
| CS2 | OOU | 27 | 0 | 0 |
| CS2 | OOUE | 18 | 0 | 0 |
| CS2 | OVAR3 | 136 | 0 | 0 |
| CS2 | VOA | 46 | 0 | 0 |
| CS3 | OOU | 18 | 0 | 0 |
| CS3 | OOUE | 11 | 0 | 0 |
| CS3 | VOA | 439 | 0 | 0 |
| CS3 | TNCO | 0 | 643 | 0 |
| CS3 | DOVE4 | 0 | 0 | 906 |

Table 3.7: Number of failed samples by CodeSet and fail condition

| CodeSet | CodeSet Total | linFlag | imagingFlag | spcFlag | normFlag | QCFlag | n |
|---|---|---|---|---|---|---|---|
| CS1 | 8 | Passed | Failed | Passed | Passed | Failed | 3 |
| | | Passed | Passed | Passed | Failed | Failed | 5 |
| CS2 | 32 | Failed | Failed | Failed | Failed | Failed | 2 |
| | | Failed | Passed | Failed | Failed | Failed | 3 |
| | | Failed | Passed | Passed | Passed | Failed | 3 |
| | | Passed | Failed | Passed | Passed | Failed | 3 |
| | | Passed | Passed | Passed | Failed | Failed | 21 |
| CS3 | 274 | Failed | Failed | Failed | Failed | Failed | 1 |
| | | Failed | Failed | Passed | Failed | Failed | 3 |
| | | Failed | Passed | Passed | Failed | Failed | 11 |
| | | Passed | Failed | Passed | Passed | Failed | 7 |
| | | Passed | Passed | Passed | Failed | Failed | 252 |

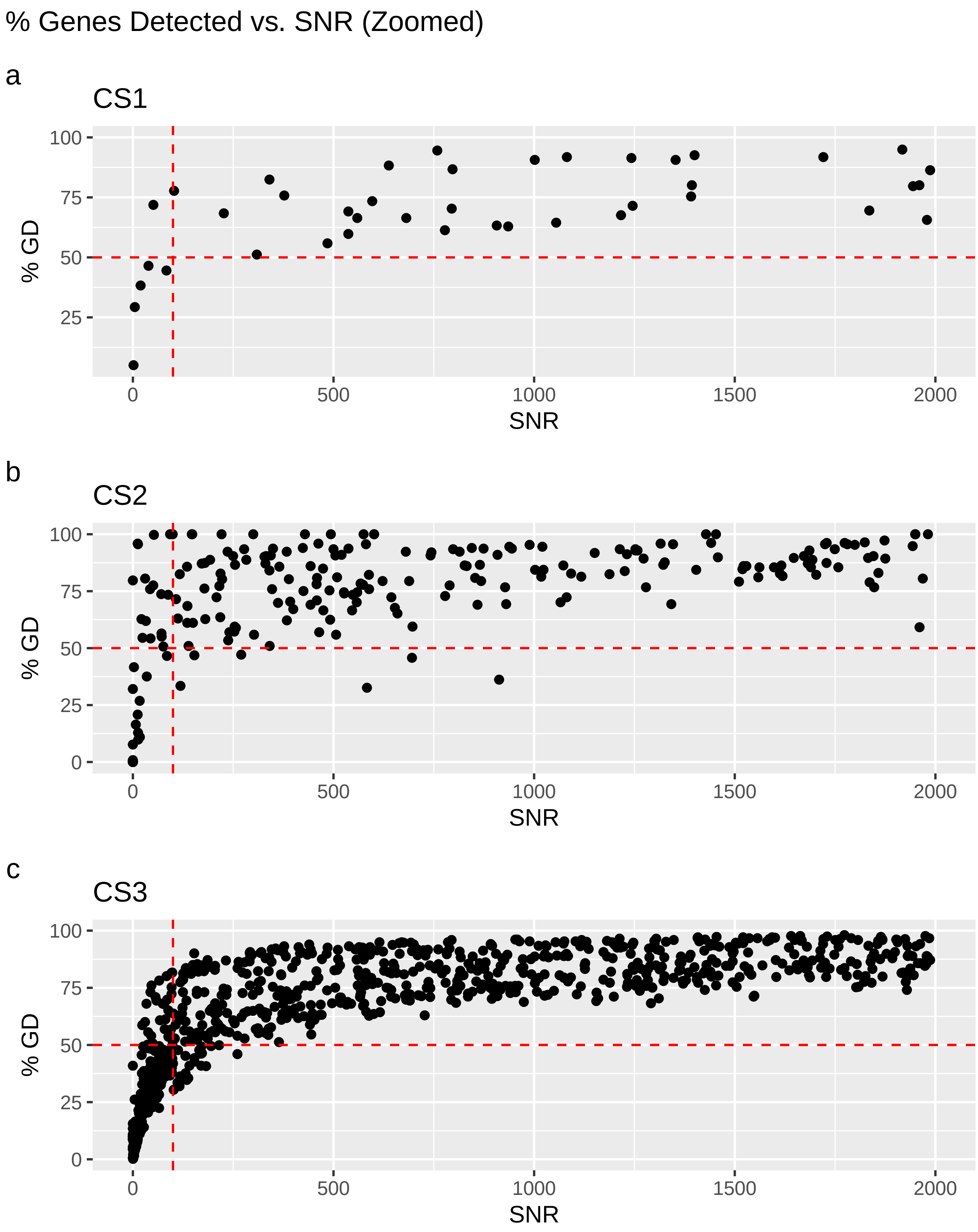


{

}

\caption{% Genes Detected vs. Signal to Noise Ratio} \end{figure}

\begin{figure}[H]



{

}

\caption{% Genes Detected vs. Signal to Noise Ratio (Zoomed)} \end{figure}

## 3.5 Pairwise Gene Expression

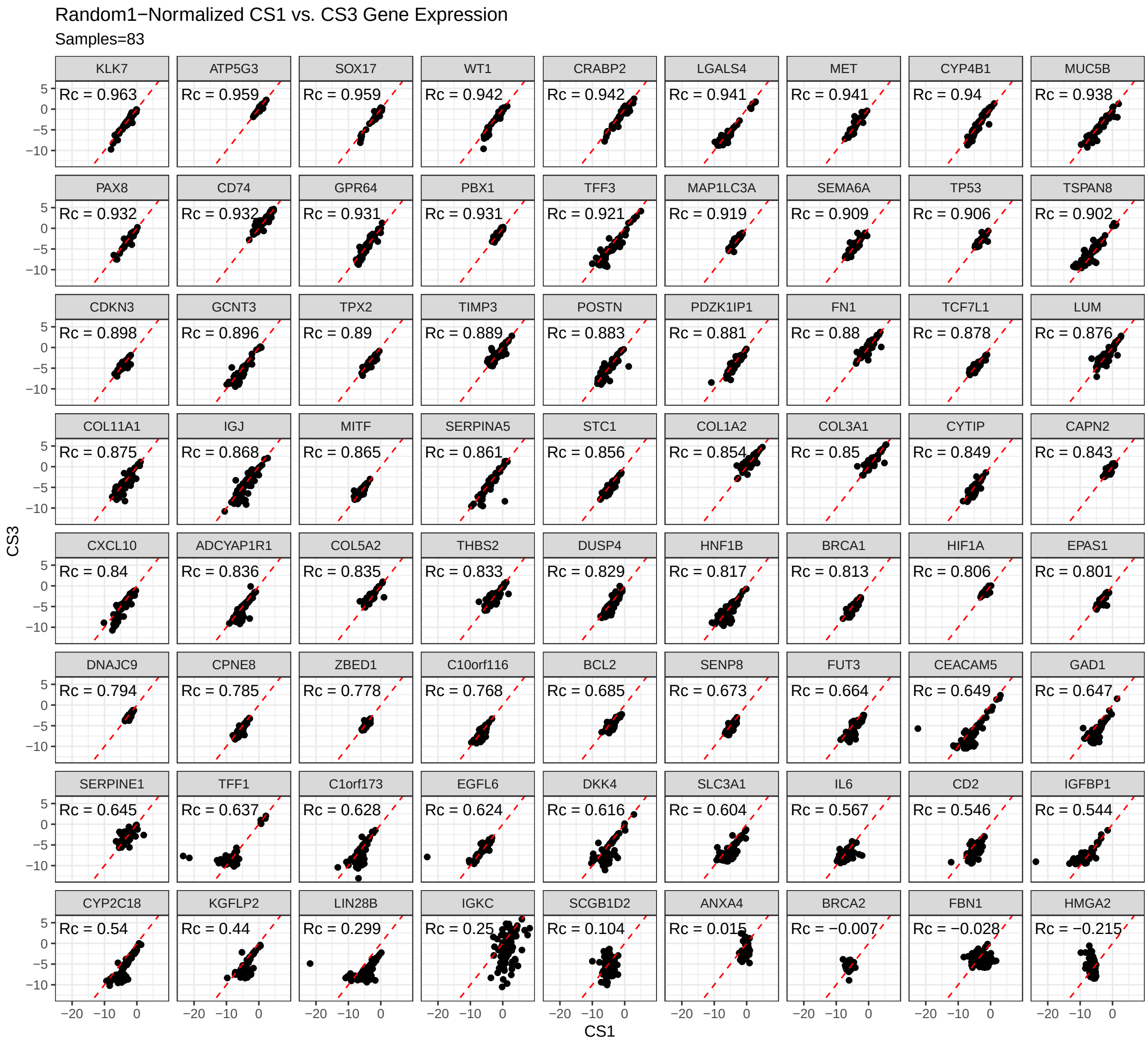


Figure 3.1: Random1-Normalized CS1 vs. CS3 Gene Expression

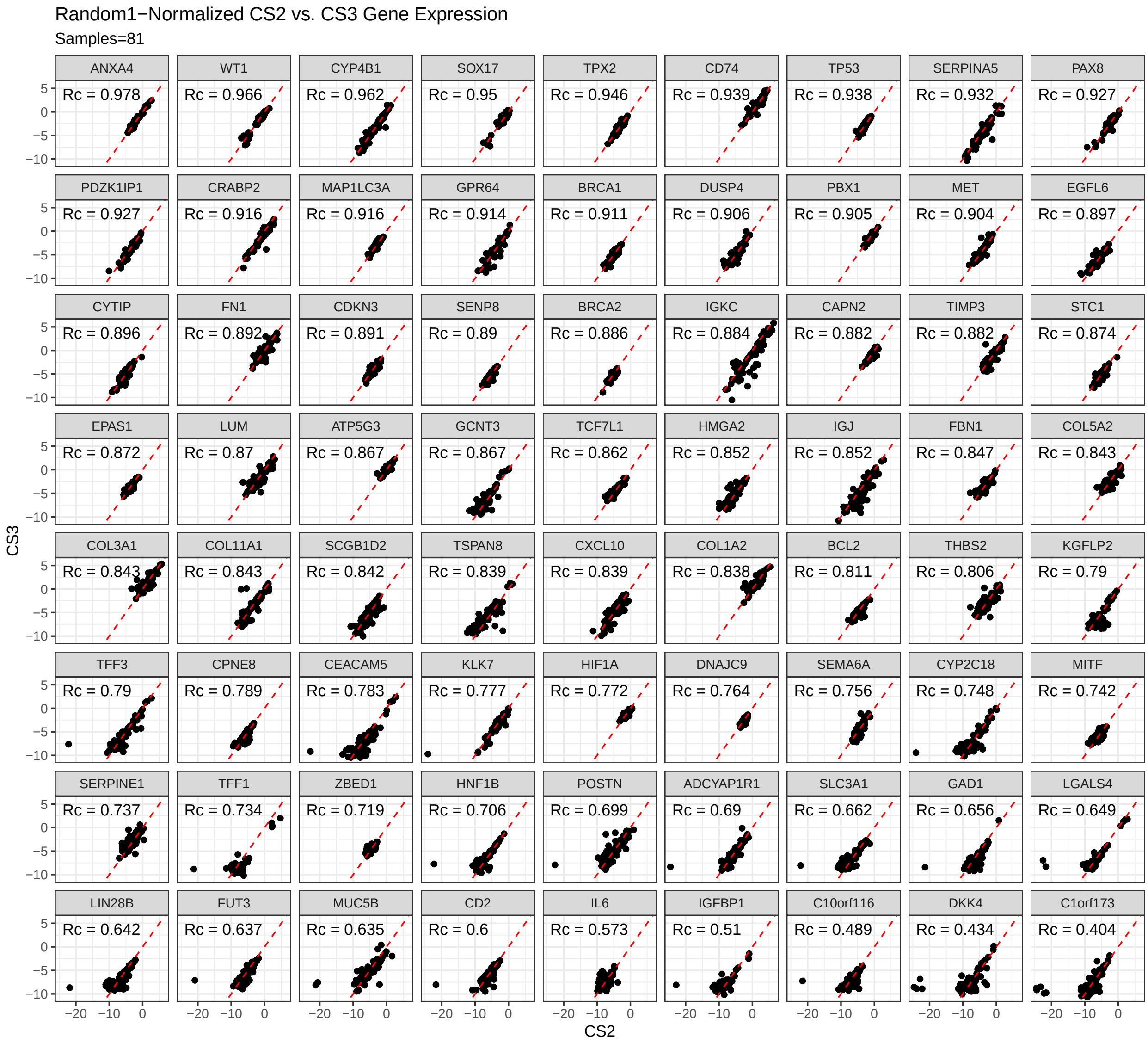


Figure 3.2: Random1-Normalized CS2 vs. CS3 Gene Expression

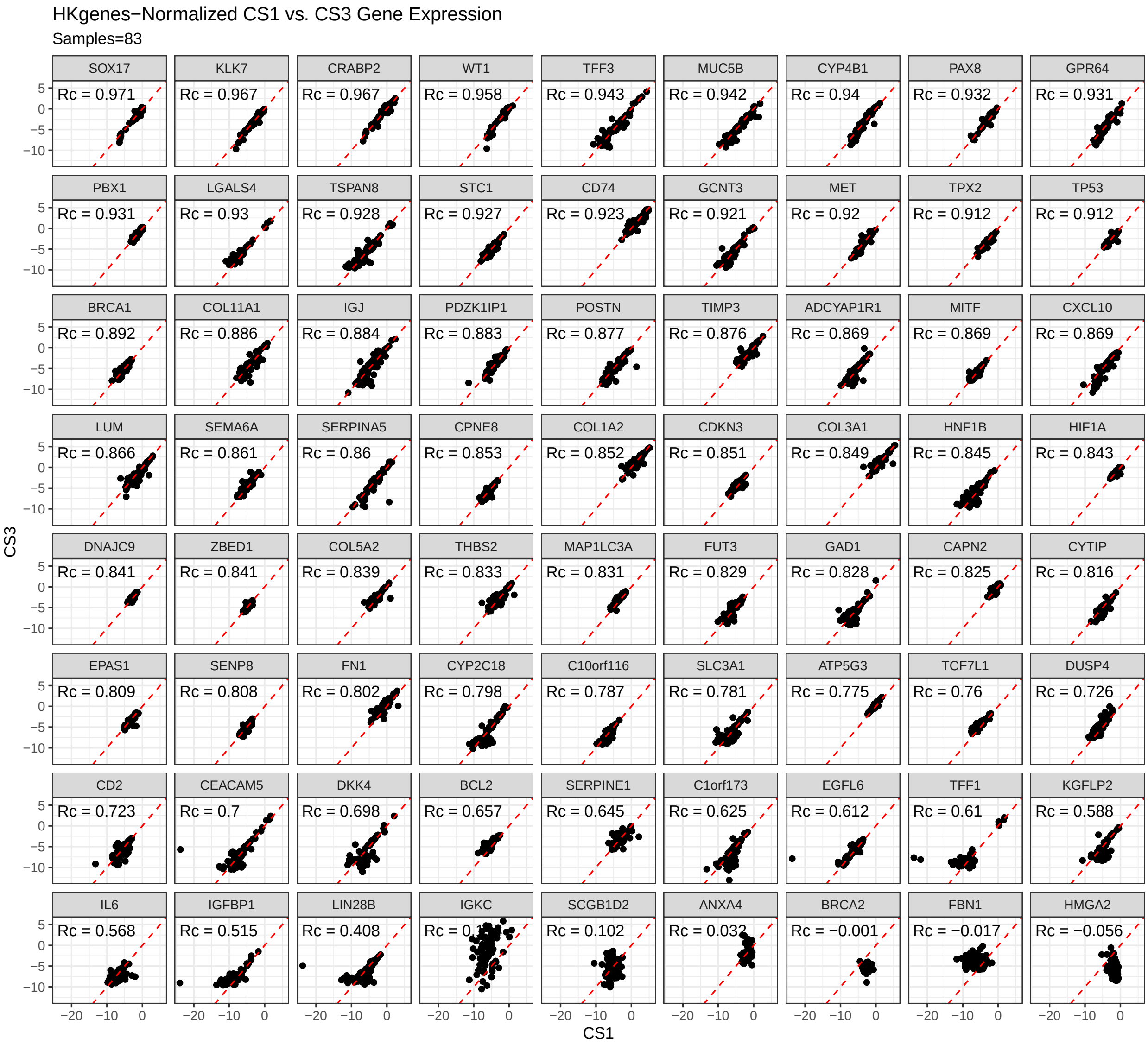


Figure 3.3: HKgenes-Normalized CS1 vs. CS3 Gene Expression

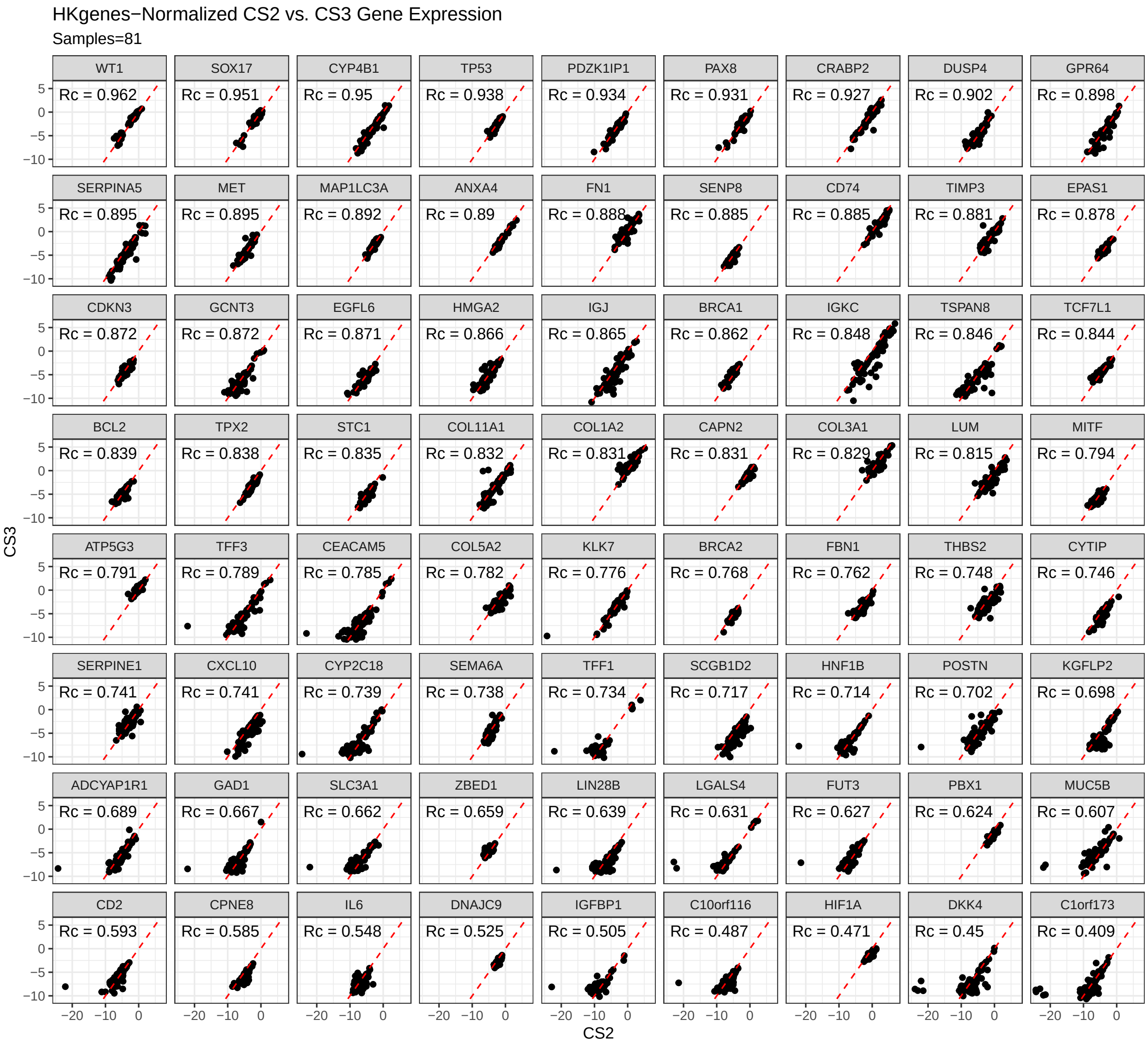


Figure 3.4: HKgenes-Normalized CS2 vs. CS3 Gene Expression

# 4. Results

We summarize cross-validated training performance of class metrics in the training set. The accuracy, F1-score, kappa, and G-mean are the metrics of interest. Workflows are ordered by their mean estimates across the outer folds of the nested CV for each metric.

## 4.1 Training Set

### 4.1.1 Accuracy

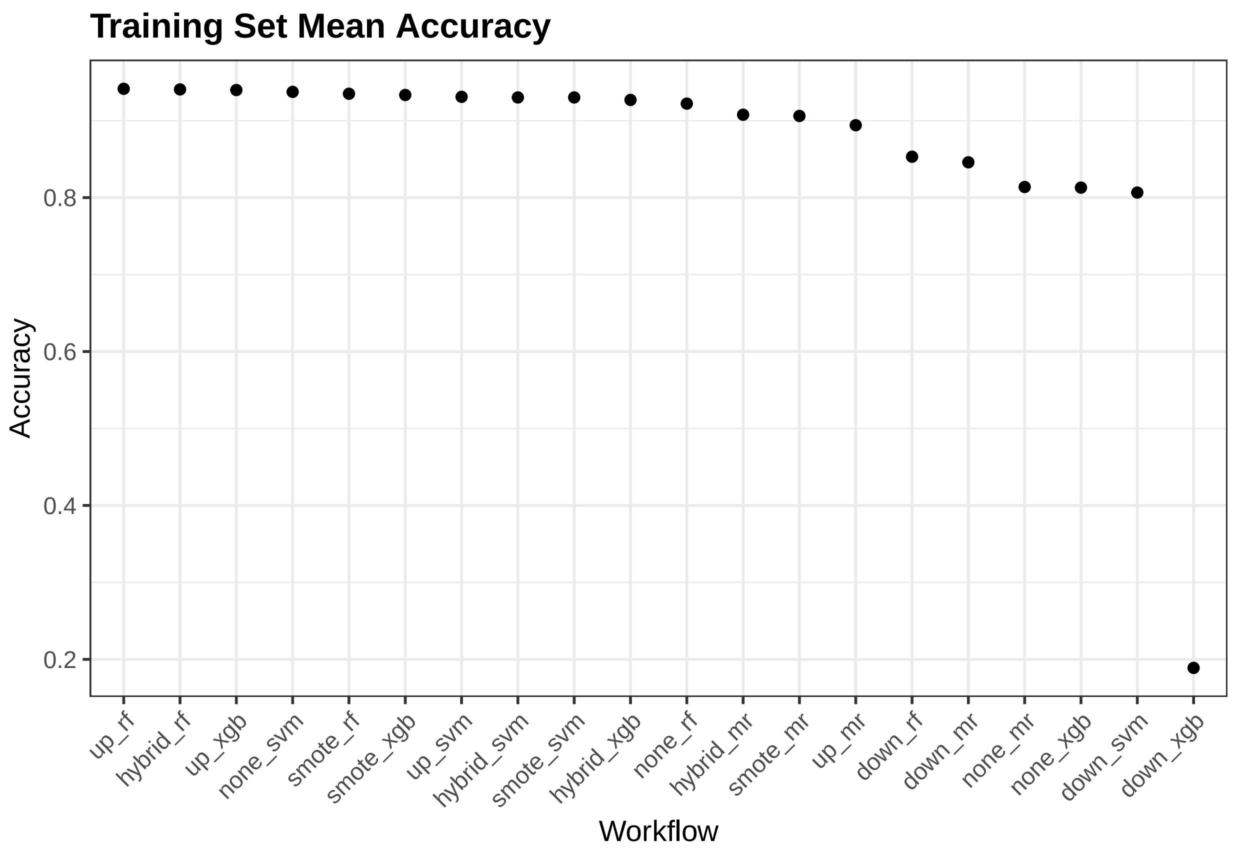


Figure 4.1: Training Set Mean Accuracy

Table 4.1: Training Set Mean Accuracy

| | Algorithms | | | |
|---|---|---|---|---|
| Subsampling | mr | rf | svm | xgb |
| none | 0.814 | 0.922 | 0.937 | 0.813 |
| down | 0.846 | 0.853 | 0.807 | 0.189 |
| up | 0.894 | **0.941** | 0.931 | 0.94 |
| smote | 0.906 | 0.935 | 0.93 | 0.933 |
| hybrid | 0.908 | **0.941** | 0.93 | 0.927 |

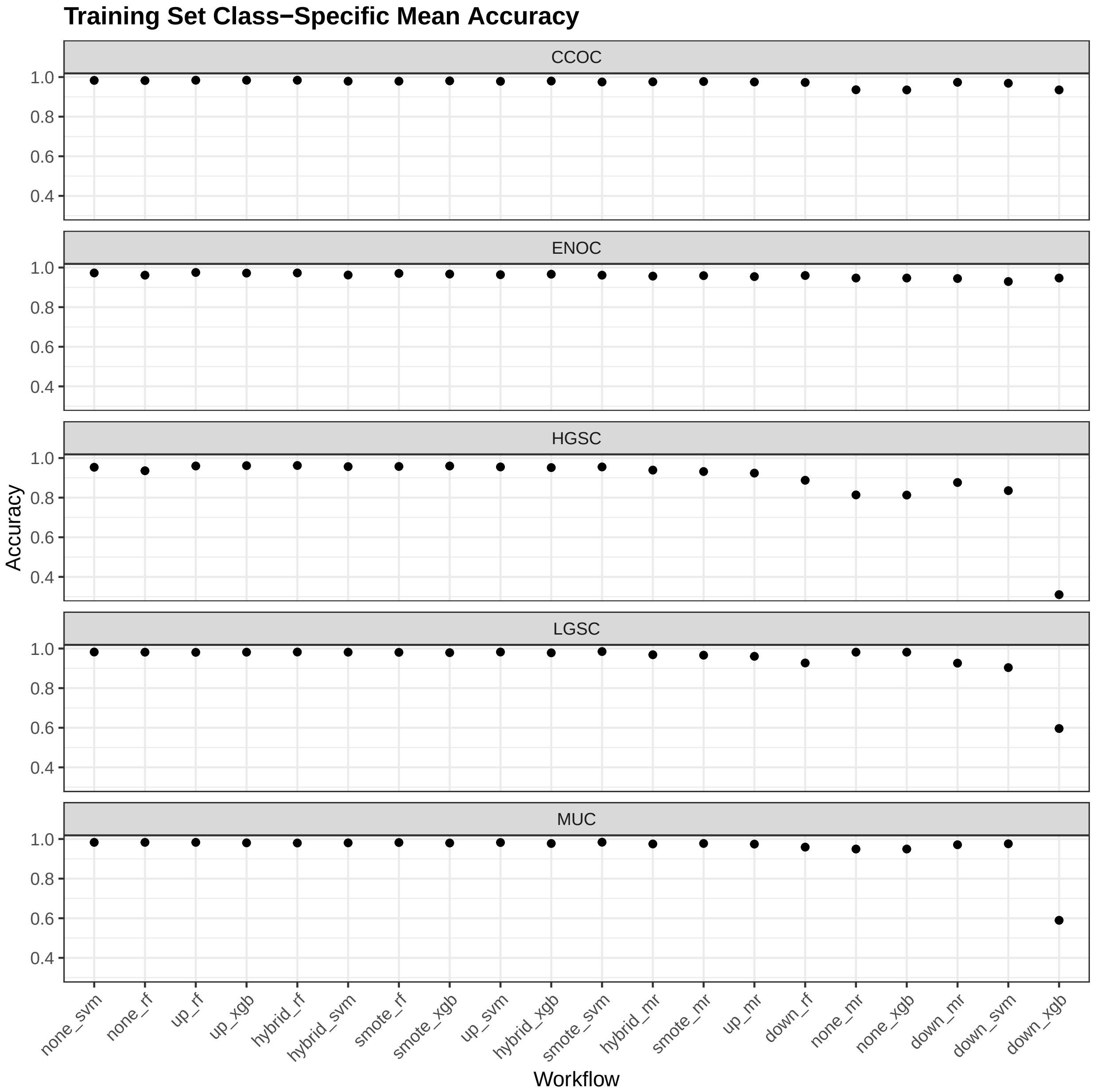


Figure 4.2: Training Set Class-Specific Mean Accuracy

Table 4.2: Training Set Class-Specific Mean Accuracy

| | | Algorithms | | | |
|---|---|---|---|---|---|
| Subsampling | Histotype | mr | rf | svm | xgb |
| none | CCOC | 0.936 | 0.982 | 0.983 | 0.935 |
| | ENOC | 0.947 | 0.961 | 0.973 | 0.947 |
| | HGSC | 0.814 | 0.936 | 0.953 | 0.813 |
| | LGSC | 0.982 | 0.982 | 0.982 | 0.982 |
| | MUC | 0.949 | 0.983 | 0.983 | 0.949 |
| down | CCOC | 0.974 | 0.973 | 0.969 | 0.935 |
| | ENOC | 0.945 | 0.96 | 0.929 | 0.947 |
| | HGSC | 0.876 | 0.888 | 0.835 | 0.31 |
| | LGSC | 0.926 | 0.927 | 0.904 | 0.596 |
| | MUC | 0.971 | 0.959 | 0.976 | 0.59 |
| up | CCOC | 0.975 | 0.984 | 0.978 | 0.984 |
| | ENOC | 0.954 | 0.975 | 0.964 | 0.972 |
| | HGSC | 0.924 | 0.96 | 0.955 | 0.961 |
| | LGSC | 0.961 | 0.981 | 0.982 | 0.982 |
| | MUC | 0.974 | 0.983 | 0.982 | 0.981 |
| smote | CCOC | 0.978 | 0.979 | 0.975 | 0.981 |
| | ENOC | 0.959 | 0.97 | 0.961 | 0.967 |
| | HGSC | 0.932 | 0.957 | 0.955 | 0.96 |
| | LGSC | 0.966 | 0.981 | **0.985** | 0.979 |
| | MUC | 0.978 | 0.982 | 0.984 | 0.98 |
| hybrid | CCOC | 0.976 | 0.984 | 0.979 | 0.98 |
| | ENOC | 0.957 | 0.973 | 0.962 | 0.966 |
| | HGSC | 0.939 | 0.962 | 0.957 | 0.952 |
| | LGSC | 0.969 | 0.982 | 0.982 | 0.978 |
| | MUC | 0.975 | 0.98 | 0.981 | 0.978 |

Table 4.3: Training Set Mean F1-Score

| | Algorithms | | | |
|---|---|---|---|---|
| Subsampling | mr | rf | svm | xgb |
| none | 0.822 | 0.787 | 0.801 | **0.897** |
| down | 0.664 | 0.657 | 0.645 | 0.231 |
| up | 0.709 | 0.755 | 0.726 | 0.776 |
| smote | 0.724 | 0.748 | 0.75 | 0.747 |
| hybrid | 0.72 | 0.77 | 0.751 | 0.728 |

### 4.1.2 F1-Score

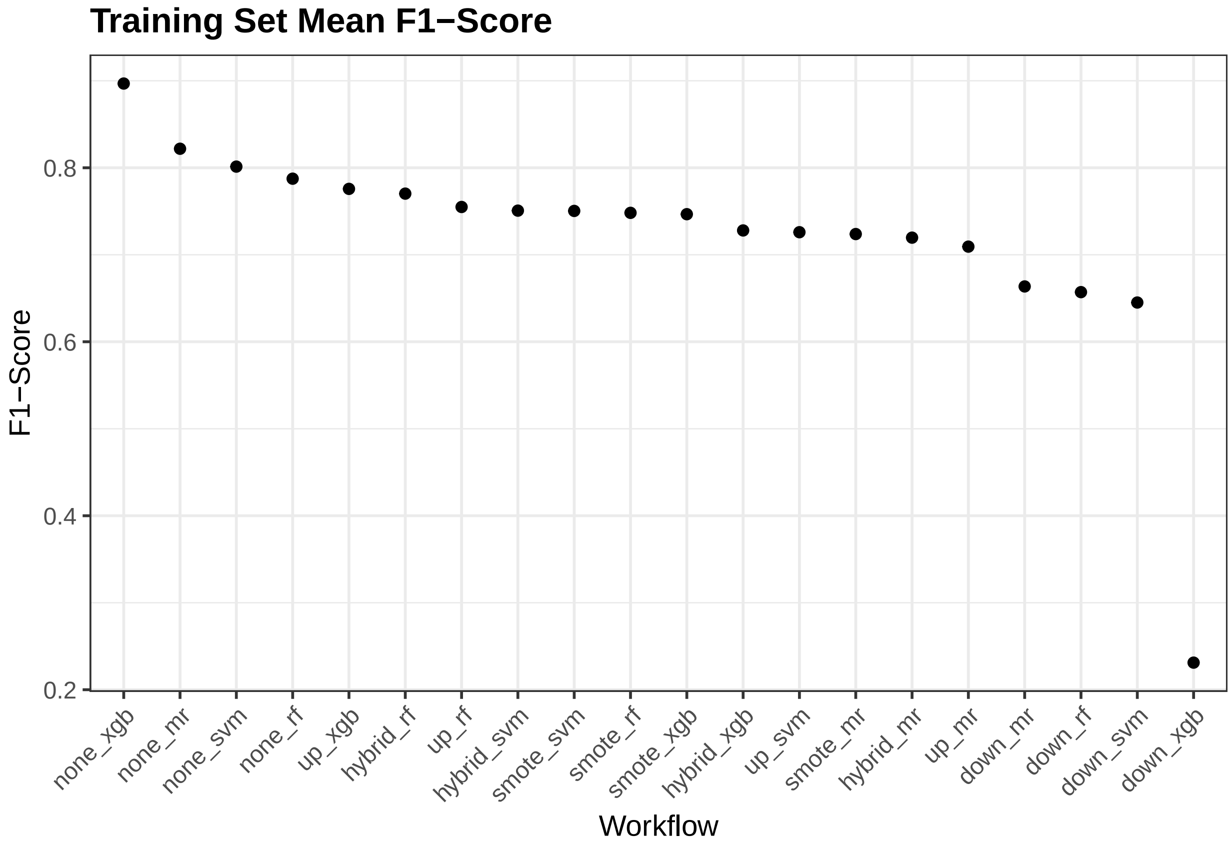


Figure 4.3: Training Set Mean F1-Score

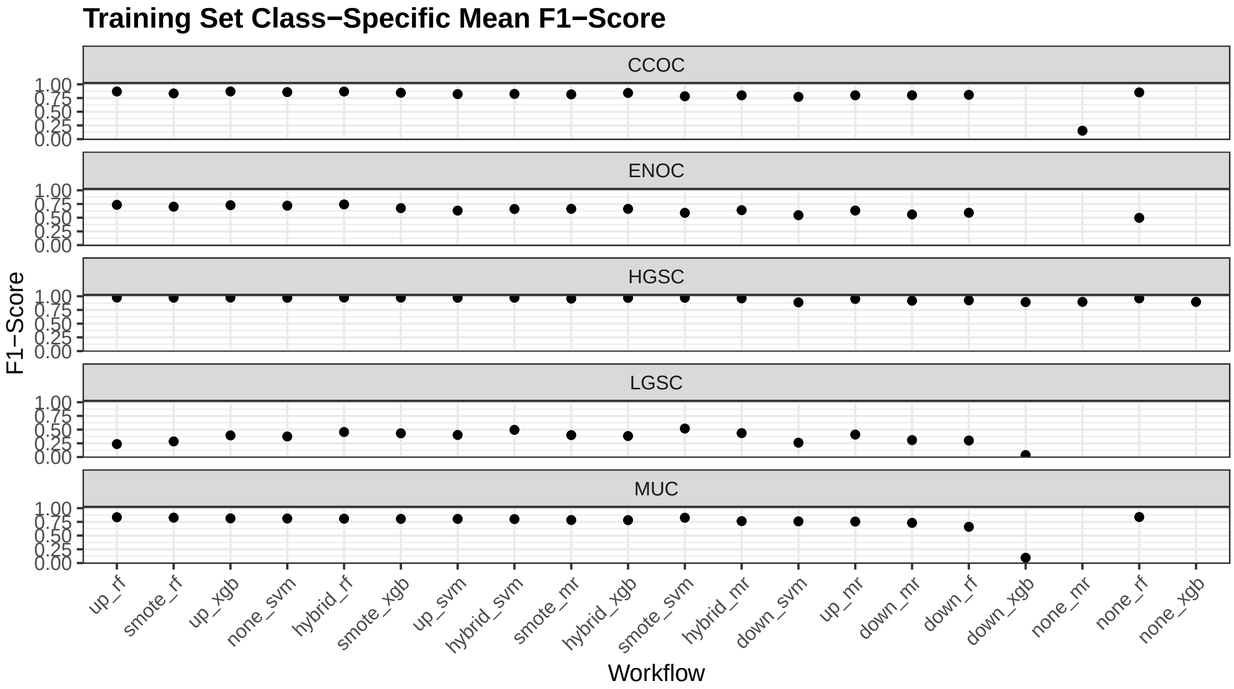


Figure 4.4: Training Set Class-Specific Mean F1-Score

Table 4.4: Cross-Validated Training Set Class-Specific Mean F1-Score

| | | Algorithms | | | |
|---|---|---|---|---|---|
| Subsampling | Histotype | mr | rf | svm | xgb |
| none | CCOC | 0.154 | 0.852 | 0.858 | NA |
| | ENOC | NA | 0.497 | 0.719 | NA |
| | HGSC | 0.897 | 0.962 | 0.972 | 0.897 |
| | LGSC | NA | NA | 0.375 | NA |
| | MUC | NA | 0.839 | 0.811 | NA |
| down | CCOC | 0.8 | 0.808 | 0.771 | NA |
| | ENOC | 0.559 | 0.59 | 0.545 | NA |
| | HGSC | 0.918 | 0.926 | 0.888 | 0.894 |
| | LGSC | 0.308 | 0.301 | 0.262 | 0.035 |
| | MUC | 0.732 | 0.66 | 0.759 | 0.096 |
| up | CCOC | 0.8 | 0.869 | 0.822 | 0.87 |
| | ENOC | 0.63 | 0.734 | 0.629 | 0.728 |
| | HGSC | 0.951 | 0.976 | 0.973 | 0.976 |
| | LGSC | 0.409 | 0.236 | 0.402 | 0.394 |
| | MUC | 0.756 | 0.835 | 0.804 | 0.814 |
| smote | CCOC | 0.817 | 0.833 | 0.782 | 0.846 |
| | ENOC | 0.662 | 0.701 | 0.588 | 0.674 |
| | HGSC | 0.957 | 0.974 | 0.973 | 0.975 |
| | LGSC | 0.4 | 0.285 | 0.519 | 0.433 |
| | MUC | 0.784 | 0.827 | 0.825 | 0.805 |
| hybrid | CCOC | 0.799 | 0.868 | 0.825 | 0.843 |
| | ENOC | 0.638 | 0.742 | 0.659 | 0.661 |
| | HGSC | 0.962 | **0.977** | 0.973 | 0.97 |
| | LGSC | 0.436 | 0.456 | 0.495 | 0.384 |
| | MUC | 0.764 | 0.808 | 0.801 | 0.782 |

Table 4.5: Training Set Mean Kappa

| | Algorithms | | | |
|---|---|---|---|---|
| Subsampling | mr | rf | svm | xgb |
| none | 0.008 | 0.727 | 0.79 | 0 |
| down | 0.63 | 0.639 | 0.569 | 0 |
| up | 0.717 | 0.81 | 0.783 | 0.814 |
| smote | 0.743 | 0.796 | 0.772 | 0.799 |
| hybrid | 0.744 | **0.819** | 0.788 | 0.783 |

### 4.1.3 Kappa

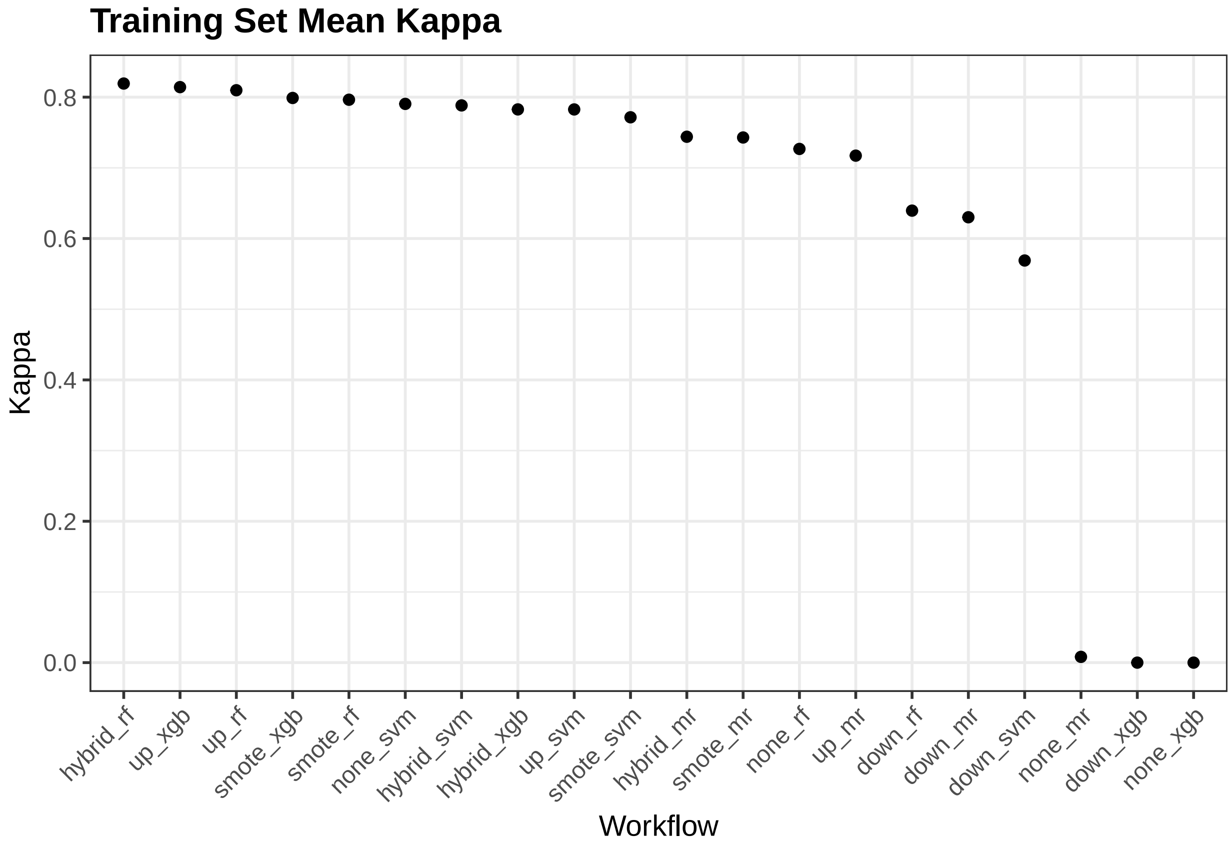


Figure 4.5: Training Set Mean Kappa

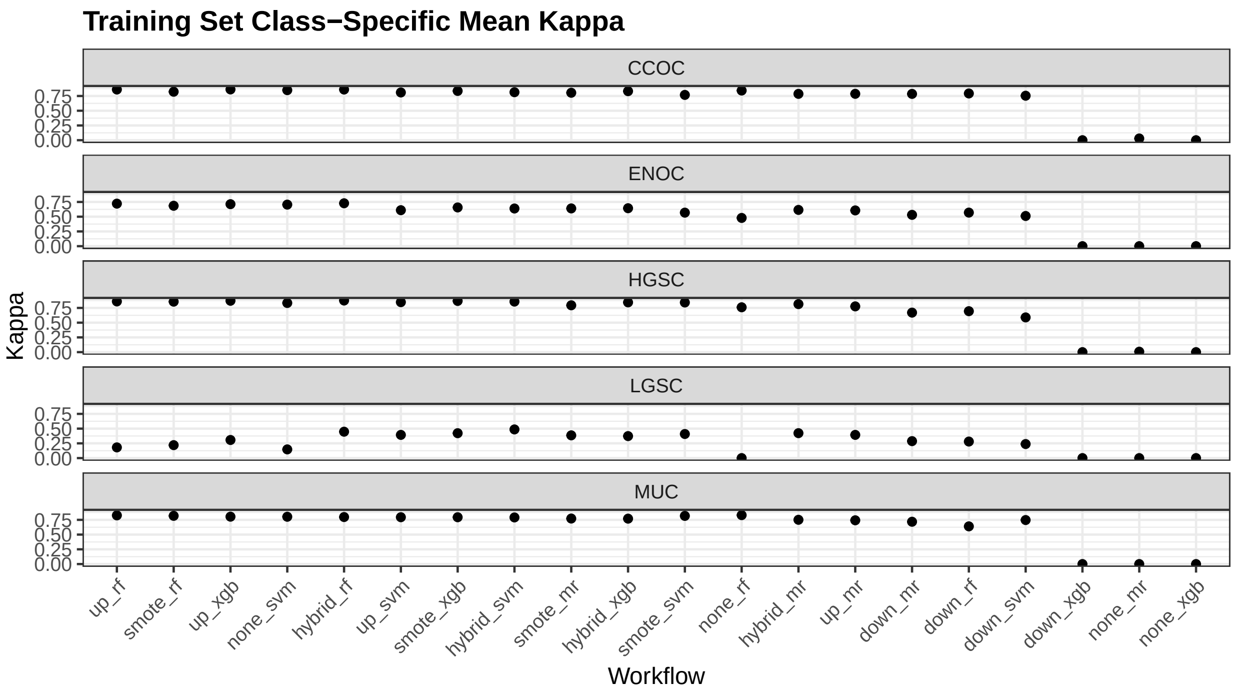


Figure 4.6: Training Set Class-Specific Mean Kappa

Table 4.6: Training Set Class-Specific Mean Kappa

| | | Algorithms | | | |
|---|---|---|---|---|---|
| Subsampling | Histotype | mr | rf | svm | xgb |
| none | CCOC | 0.03 | 0.843 | 0.849 | 0 |
| | ENOC | 0 | 0.479 | 0.705 | 0 |
| | HGSC | 0.008 | 0.761 | 0.834 | 0 |
| | LGSC | 0 | 0 | 0.148 | 0 |
| | MUC | 0 | 0.83 | 0.803 | 0 |
| down | CCOC | 0.786 | 0.793 | 0.755 | 0 |
| | ENOC | 0.531 | 0.57 | 0.512 | 0 |
| | HGSC | 0.671 | 0.695 | 0.589 | 0 |
| | LGSC | 0.288 | 0.281 | 0.238 | 0 |
| | MUC | 0.718 | 0.64 | 0.747 | 0 |
| up | CCOC | 0.787 | 0.86 | 0.81 | 0.861 |
| | ENOC | 0.607 | 0.721 | 0.61 | 0.713 |
| | HGSC | 0.776 | 0.86 | 0.848 | 0.872 |
| | LGSC | 0.394 | 0.181 | 0.394 | 0.307 |
| | MUC | 0.743 | 0.826 | 0.795 | 0.804 |
| smote | CCOC | 0.805 | 0.822 | 0.769 | 0.836 |
| | ENOC | 0.64 | 0.685 | 0.569 | 0.657 |
| | HGSC | 0.795 | 0.856 | 0.842 | 0.869 |
| | LGSC | 0.386 | 0.22 | 0.409 | 0.422 |
| | MUC | 0.772 | 0.818 | 0.817 | 0.795 |
| hybrid | CCOC | 0.786 | 0.86 | 0.814 | 0.832 |
| | ENOC | 0.615 | 0.728 | 0.639 | 0.644 |
| | HGSC | 0.815 | **0.876** | 0.857 | 0.844 |
| | LGSC | 0.423 | 0.448 | 0.486 | 0.373 |
| | MUC | 0.751 | 0.798 | 0.791 | 0.771 |

Table 4.7: Training Set Mean G-mean

| | Algorithms | | | |
|---|---|---|---|---|
| Subsampling | mr | rf | svm | xgb |
| none | 0.858 | 0.705 | 0.732 | **1** |
| down | 0.821 | 0.786 | 0.837 | **1** |
| up | 0.817 | 0.717 | 0.722 | 0.735 |
| smote | 0.847 | 0.739 | 0.694 | 0.738 |
| hybrid | 0.803 | 0.729 | 0.739 | 0.769 |

### 4.1.4 G-mean

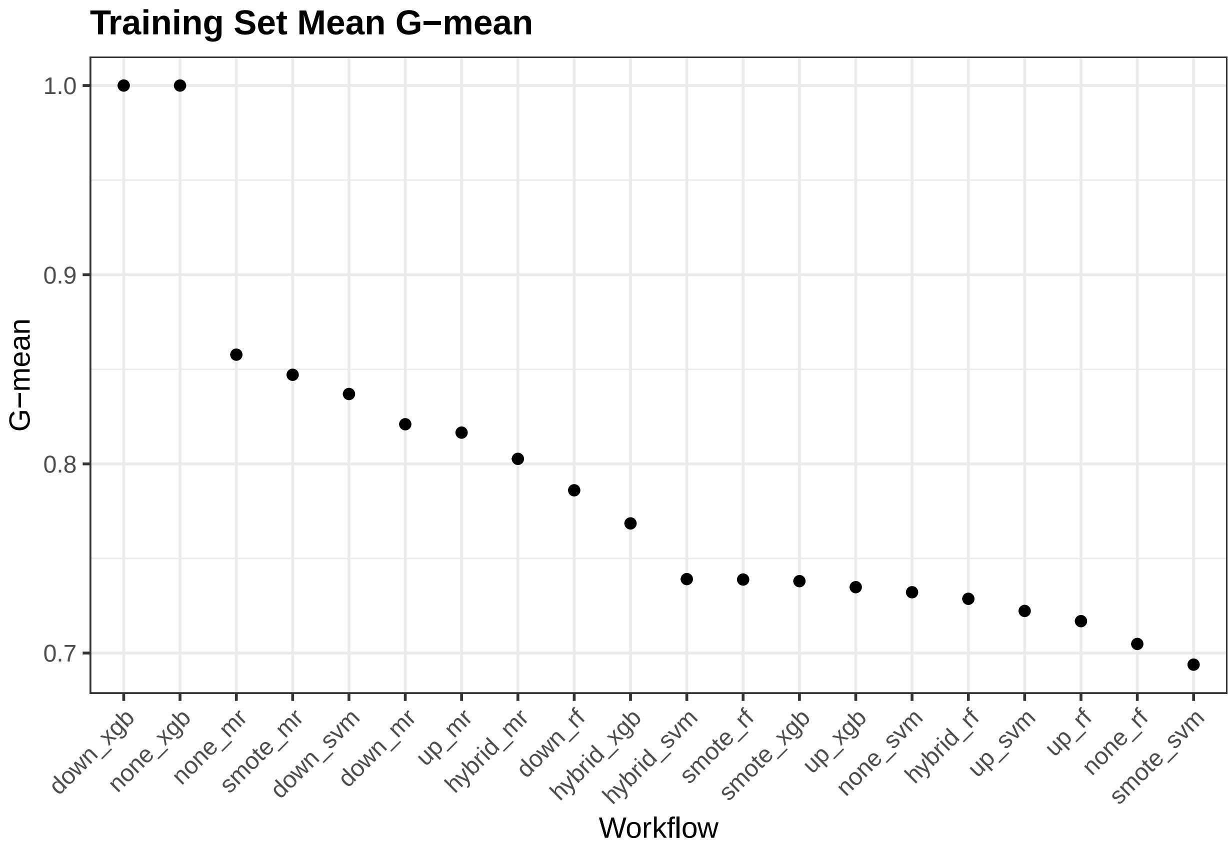


Figure 4.7: Training Set Mean G-mean

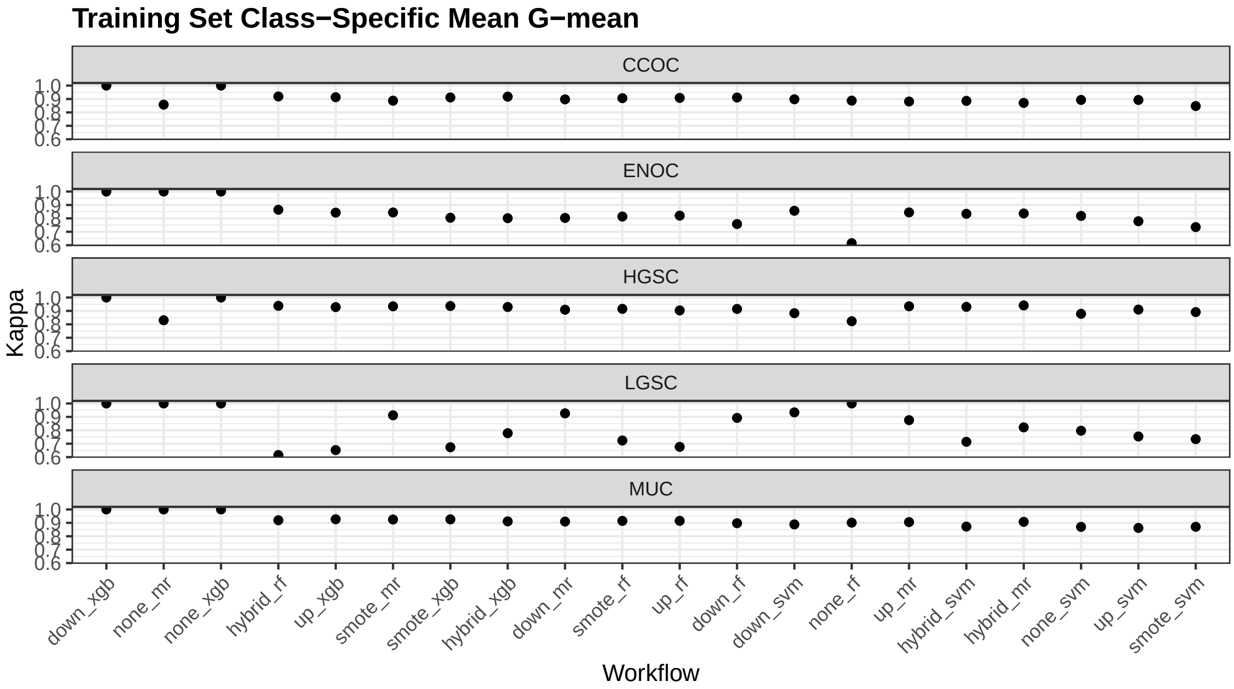


Figure 4.8: Training Set Class-Specific Mean G-mean

Table 4.8: Training Set Class-Specific Mean G-mean

| | | Algorithms | | | |
|---|---|---|---|---|---|
| Subsampling | Histotype | mr | rf | svm | xgb |
| none | CCOC | 0.858 | 0.888 | 0.893 | **1** |
| | ENOC | **1** | 0.615 | 0.818 | **1** |
| | HGSC | 0.83 | 0.823 | 0.878 | **1** |
| | LGSC | **1** | **1** | 0.797 | **1** |
| | MUC | **1** | 0.901 | 0.87 | **1** |
| down | CCOC | 0.897 | 0.911 | 0.898 | **1** |
| | ENOC | 0.803 | 0.758 | 0.857 | **1** |
| | HGSC | 0.909 | 0.915 | 0.883 | **1** |
| | LGSC | 0.926 | 0.892 | 0.934 | **1** |
| | MUC | 0.909 | 0.897 | 0.888 | **1** |
| up | CCOC | 0.881 | 0.908 | 0.893 | 0.913 |
| | ENOC | 0.844 | 0.82 | 0.779 | 0.843 |
| | HGSC | 0.935 | 0.904 | 0.91 | 0.928 |
| | LGSC | 0.876 | 0.677 | 0.754 | 0.653 |
| | MUC | 0.906 | 0.915 | 0.862 | 0.927 |
| smote | CCOC | 0.887 | 0.906 | 0.847 | 0.912 |
| | ENOC | 0.844 | 0.814 | 0.735 | 0.805 |
| | HGSC | 0.934 | 0.915 | 0.891 | 0.937 |
| | LGSC | 0.912 | 0.724 | 0.734 | 0.673 |
| | MUC | 0.925 | 0.914 | 0.87 | 0.926 |
| hybrid | CCOC | 0.871 | 0.919 | 0.886 | 0.917 |
| | ENOC | 0.837 | 0.864 | 0.834 | 0.802 |
| | HGSC | 0.941 | 0.938 | 0.93 | 0.929 |
| | LGSC | 0.822 | 0.615 | 0.714 | 0.779 |
| | MUC | 0.908 | 0.919 | 0.872 | 0.911 |

## 4.2 Optimal Gene Sets

### 4.2.1 Sequential Algorithm

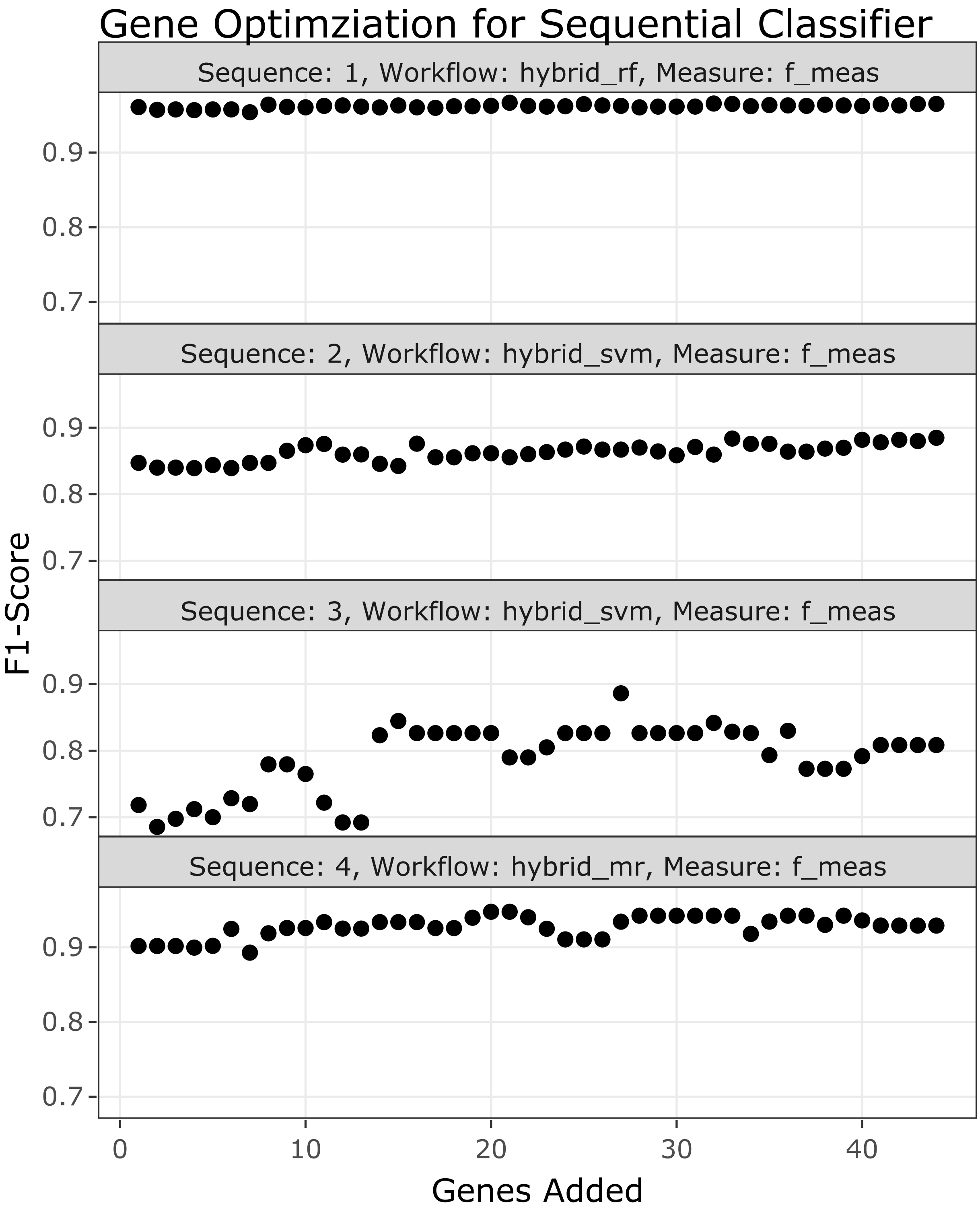


Figure 4.9: Gene Optimization for Sequential Classifier

In the sequential algorithm, sequences 1, 2, and 4 have relatively flat average F1-scores across the number of genes added. However, we can observe in sequence 3, the F1-score stabilizes at around 0.9 when we reach 27 genes added, hence the optimal number of genes used will be n=28+27=55 The added genes are: CYP2C18, HNF1B, ATP5G3, TP53, SLC3A1, CPNE8, C1orf173, WT1, MUC5B, MAP1LC3A, EGFL6, ZBED1, GPR64, STC1, MET, IGJ, SERPINA5, KLK7, DKK4, BCL2, SENP8, GCNT3, IGKC, IGFBP1, CAPN2, GAD1 and SCGB1D2.

### 4.2.2 Two-Step Algorithm

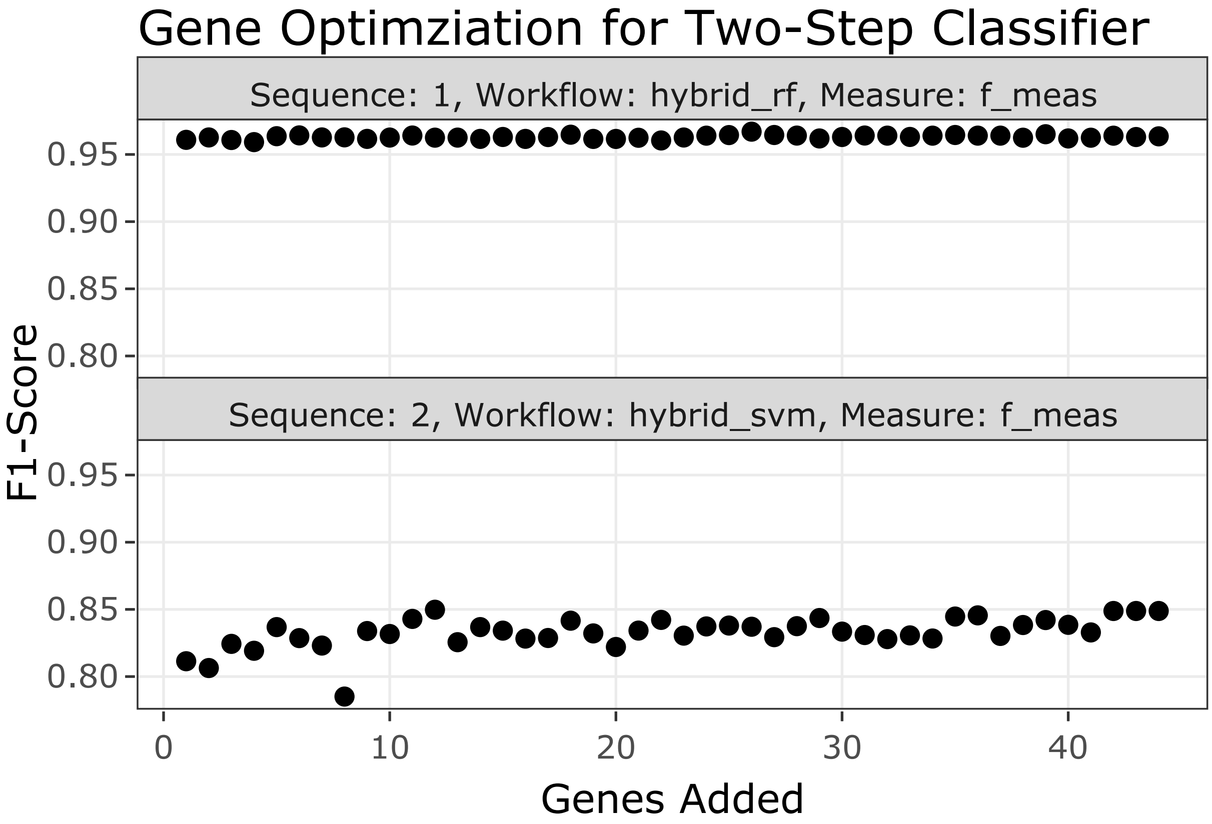
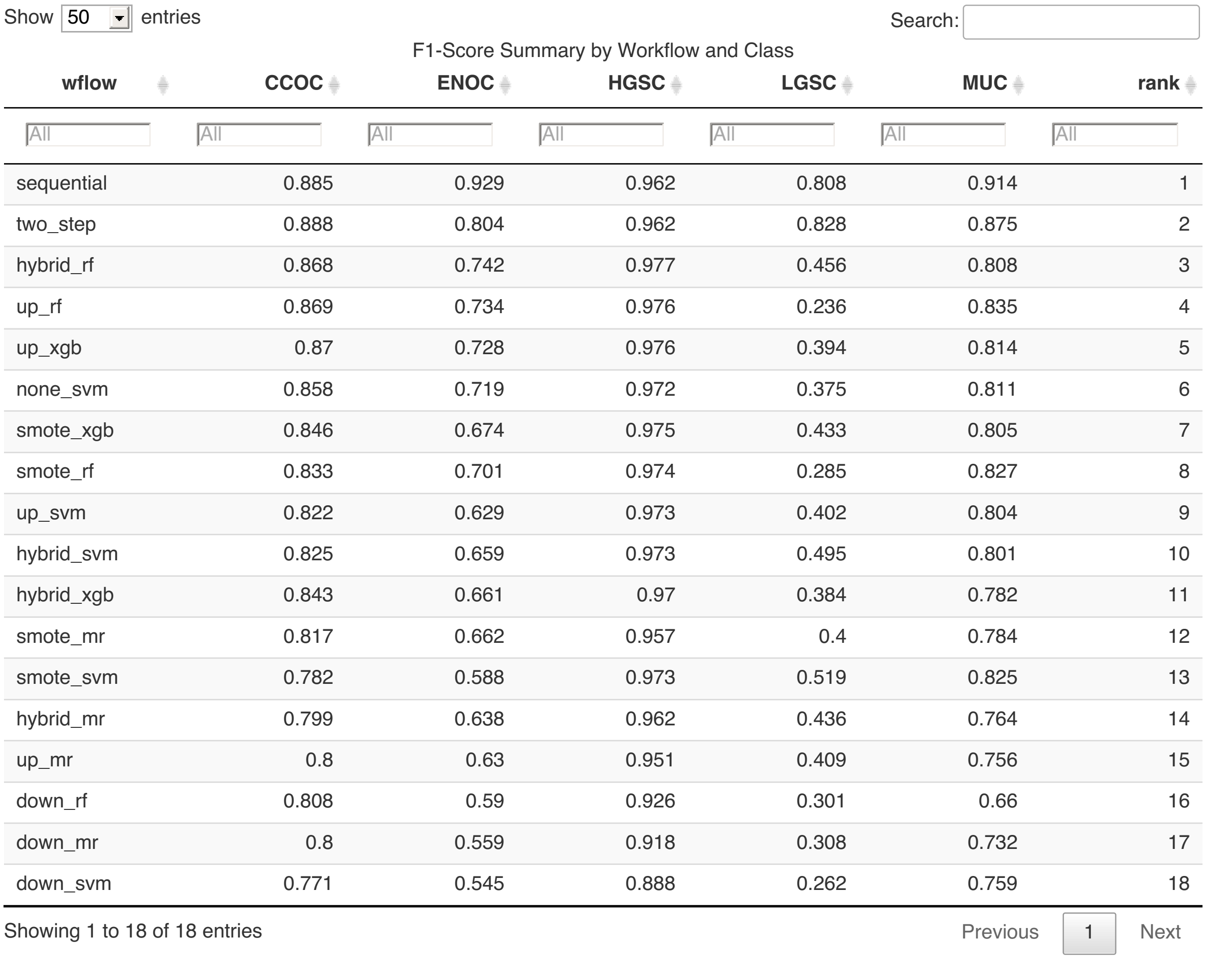


Figure 4.10: Gene Optimization for Two-Step Classifier

Since the second step of the classifier fits a multinomial model, we use the macro F1-score as the measure to analyze gene entry. In the two-step classifier, we see that in Step 2, the F1-score stabilizes at around 0.85 when we reach 12 added. The optimal number of genes used will be n=28+12=40. The added genes are: CYP2C18, MUC5B, HNF1B, SLC3A1, WT1, TSPAN8, EGFL6, TFF1, TFF3, MET, CAPN2 and KLK7.

## 4.3 Rank Aggregation



F1-Score Summary by Workflow and Class

| wflow | CCOC | ENOC | HGSC | LGSC | MUC | rank |
|---|---|---|---|---|---|---|
| All | All | All | All | All | All | All |
| sequential | 0.885 | 0.929 | 0.962 | 0.808 | 0.914 | 1 |
| two_step | 0.888 | 0.804 | 0.962 | 0.828 | 0.875 | 2 |
| hybrid_rf | 0.868 | 0.742 | 0.977 | 0.456 | 0.808 | 3 |
| up_rf | 0.869 | 0.734 | 0.976 | 0.236 | 0.835 | 4 |
| up_xgb | 0.87 | 0.728 | 0.976 | 0.394 | 0.814 | 5 |
| none_svm | 0.858 | 0.719 | 0.972 | 0.375 | 0.811 | 6 |
| smote_xgb | 0.846 | 0.674 | 0.975 | 0.433 | 0.805 | 7 |
| smote_rf | 0.833 | 0.701 | 0.974 | 0.285 | 0.827 | 8 |
| up_svm | 0.822 | 0.629 | 0.973 | 0.402 | 0.804 | 9 |
| hybrid_svm | 0.825 | 0.659 | 0.973 | 0.495 | 0.801 | 10 |
| hybrid_xgb | 0.843 | 0.661 | 0.97 | 0.384 | 0.782 | 11 |
| smote_mr | 0.817 | 0.662 | 0.957 | 0.4 | 0.784 | 12 |
| smote_svm | 0.782 | 0.588 | 0.973 | 0.519 | 0.825 | 13 |
| hybrid_mr | 0.799 | 0.638 | 0.962 | 0.436 | 0.764 | 14 |
| up_mr | 0.8 | 0.63 | 0.951 | 0.409 | 0.756 | 15 |
| down_rf | 0.808 | 0.59 | 0.926 | 0.301 | 0.66 | 16 |
| down_mr | 0.8 | 0.559 | 0.918 | 0.308 | 0.732 | 17 |
| down_svm | 0.771 | 0.545 | 0.888 | 0.262 | 0.759 | 18 |

Showing 1 to 18 of 18 entries

The 18 workflows are ordered in the table by their aggregated ranks using the Genetic Algorithm. We see that the best performing methods involve the sequential and two-step algorithms.

### 4.3.1 Top Workflows

We look at the per-class evaluation metrics of the top 4 workflows.

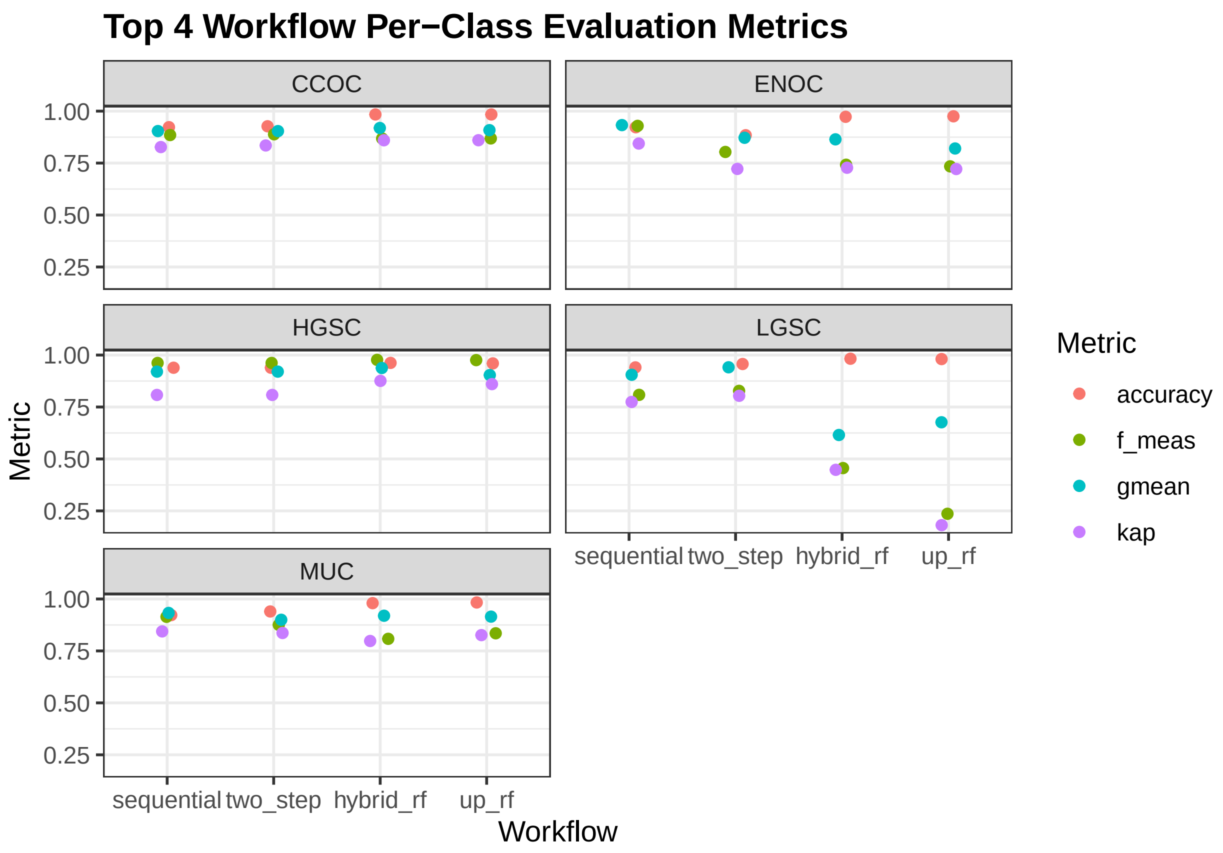


Figure 4.11: Top 4 Workflow Per-Class Evaluation Metrics

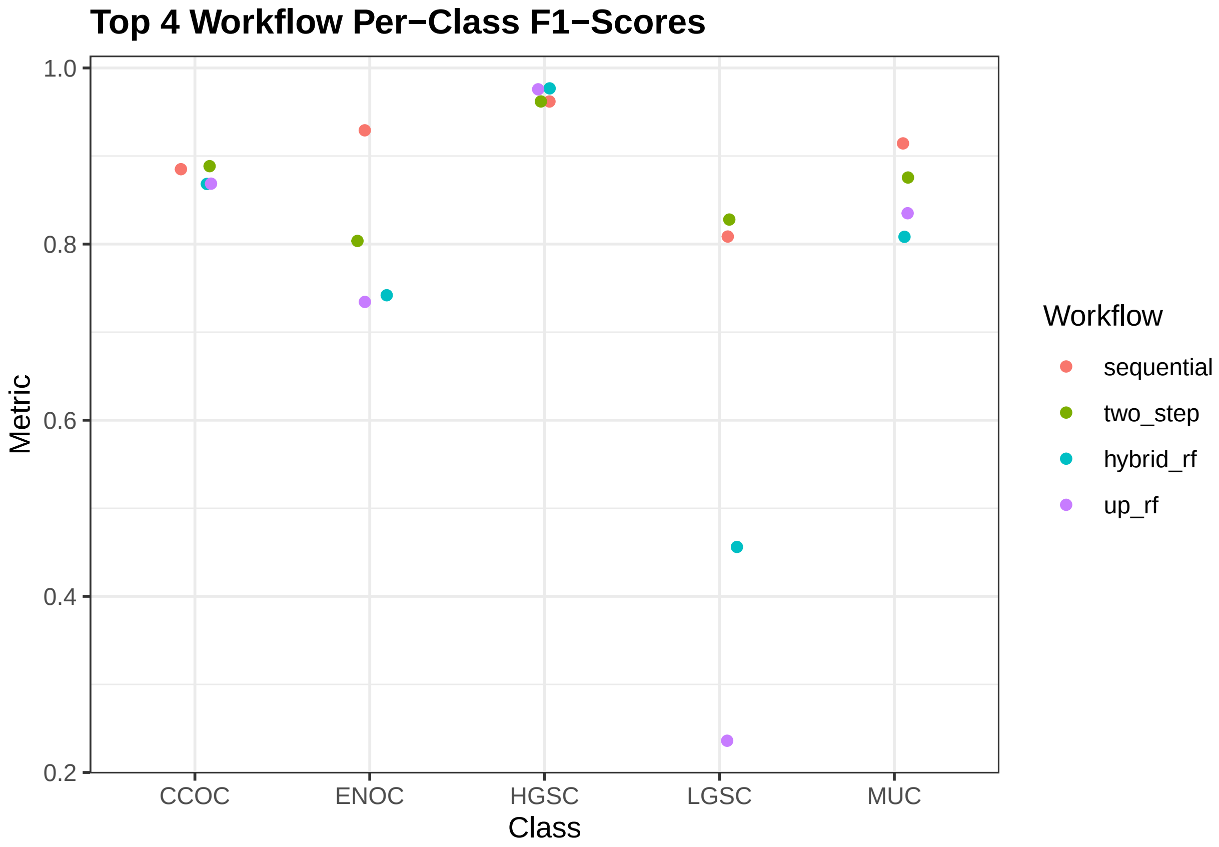


Figure 4.12: Top 4 Workflow Per-Class F1-Scores

Misclassified cases from a previous step of the sequence of classifiers are not included in subsequent steps of the training set CV folds. Thus, we cannot piece together the test set predictions from the sequential and two-step algorithms to obtain overall metrics.

## 4.4 Test Set Performance

Now we'd like to see how our best methods perform in the confirmation and validation sets. The class-specific F1-scores will be used.

The top 2 methods are:

- **sequential**: sequential algorithm with hybrid subsampling at every step. The sequence of algorithms used are:
  - HGSC vs. non-HGSC using random forest
  - CCOC vs. non-CCOC using support vector machine
  - LGSC vs. non-LGSC using support vector machine
  - ENOC vs. MUC using regularized multinomial regression
- **two_step**: two-step algorithm with hybrid subsampling at both steps. The sequence of algorithms used are:
  - HGSC vs. non-HGSC using random forest

Table 4.9: Overall Evaluation Metrics on Confirmation Set Models

| method | accuracy | kappa | f1 | gmean |
|---|---|---|---|---|
| sequential_full | 0.834 | 0.669 | 0.654 | 0.574 |
| sequential_optimal | 0.827 | 0.660 | 0.642 | 0.574 |
| two_step_full | 0.840 | 0.682 | 0.688 | 0.650 |
| two_step_optimal | 0.834 | 0.668 | 0.679 | 0.639 |

Table 4.10: Per-Class Eevaluation Metrics on Confirmation Set Model

| method | .metric | CCOC | ENOC | HGSC | LGSC | MUC |
|---|---|---|---|---|---|---|
| two_step_full | accuracy | 0.970 | 0.896 | 0.869 | 0.969 | 0.975 |
| | f_meas | 0.872 | 0.626 | 0.904 | 0.333 | 0.704 |
| | kap | 0.856 | 0.568 | 0.701 | 0.318 | 0.691 |
| | gmean | 0.924 | 0.715 | 0.833 | 0.614 | 0.833 |
| two_step_optimal | accuracy | 0.969 | 0.890 | 0.866 | 0.964 | 0.978 |
| | f_meas | 0.865 | 0.594 | 0.902 | 0.303 | 0.731 |
| | kap | 0.847 | 0.534 | 0.692 | 0.286 | 0.719 |
| | gmean | 0.916 | 0.689 | 0.827 | 0.613 | 0.835 |
| sequential_full | accuracy | 0.961 | 0.893 | 0.869 | 0.969 | 0.975 |
| | f_meas | 0.839 | 0.619 | 0.904 | 0.231 | 0.680 |
| | kap | 0.817 | 0.558 | 0.701 | 0.215 | 0.667 |
| | gmean | 0.919 | 0.714 | 0.833 | 0.477 | 0.790 |
| sequential_optimal | accuracy | 0.950 | 0.896 | 0.869 | 0.967 | 0.972 |
| | f_meas | 0.800 | 0.617 | 0.903 | 0.222 | 0.667 |
| | kap | 0.772 | 0.560 | 0.702 | 0.206 | 0.652 |
| | gmean | 0.907 | 0.704 | 0.836 | 0.476 | 0.811 |

- CCOC vs. ENOC vs. MUC vs. LGSC support vector machine

We can test 2 additional methods by using either the full set of genes or the optimal set of genes for both of these methods.

### 4.4.1 Confirmation Set

### 4.4.2 Validation Set

Table 4.11: Overall Evaluation Metrics on Validation Set Model

| method | accuracy | kappa | f1 | gmean |
|---|---|---|---|---|
| two_step_optimal | 0.847 | 0.653 | 0.65 | 0.701 |

Table 4.12: Per-Class Eevaluation Metrics on Validation Set Model

| method | .metric | CCOC | ENOC | HGSC | LGSC | MUC |
|---|---|---|---|---|---|---|
| two_step_optimal | accuracy | 0.972 | 0.922 | 0.879 | 0.964 | 0.957 |
| | f_meas | 0.854 | 0.608 | 0.917 | 0.353 | 0.519 |
| | kap | 0.839 | 0.566 | 0.690 | 0.336 | 0.499 |
| | gmean | 0.945 | 0.706 | 0.853 | 0.697 | 0.881 |