Predicting pathological Complete Response

Simon Ma

Table of Contents

1. Overview
2. Exploratory Data Analysis
3. Feature Selection
4. Baseline Model: Logistic Regression
5. Model II: Random Forest
6. Model III: Gradient Boosting Machine
7. Interpretation
8. Limitations
9. Appendix

Project Overview

Evaluation of pam50 predictor against others

Prosigna predicted pCR using categorical pam50 predictor

We use metric to compare our classifiers against pam50 benchmark

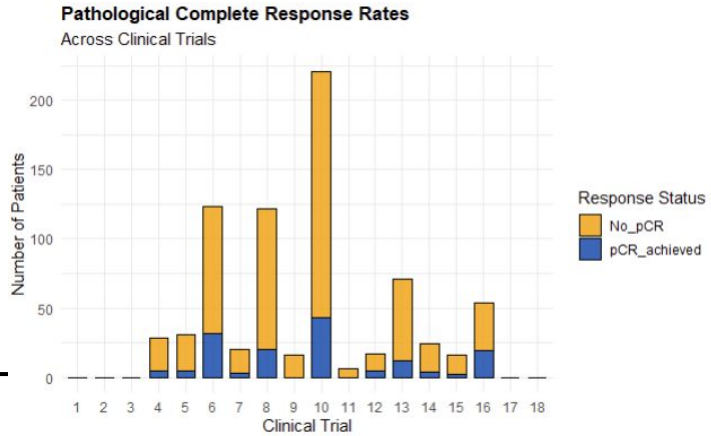
Analysis Workflow

1. **Data Exploration**: Examine distributions and relationships
2. **Feature Selection**: Identify informative biomarkers
3. **Model Training**: Train classification model using selected features
4. **Validation**: Cross-validate across multiple studies
5. **Benchmarking**: Compare performance against pam50 predictor
6. **Interpretation**: Extract biological insights from predictive feature

Exploratory Data Analysis: Structure

Overview

- **pCR Data**
  - Access study 1 responses: pCR[[1]]
  - Binary values (0 or 1)
- **Expression Data**
  - Access study 1 biomarkers: XX_pCR[[1]]
  - Matrix dimensions: 10,115 biomarkers × [variable # of patients]
  - Row names = biomarker identifiers



Objects

- **Data Source**: 18 studies with binary pCR outcomes
- **Three Main Components**:
  - pCR: List of 18 binary response vectors
  - XX_pCR: List of 18 expression matrices (biomarkers × patients)
  - pam50_recur: Benchmark prediction vectors
- **Scale**: 10,115 biomarkers consistent across all studies

Test and Train Sets:

- Study 10 selected as the **test set** (largest dataset with pam50 predictions, minimizing evaluation variance)
- Remaining studies with pam50 predictions combined as the **training set**

Cross-Validation Approach:

- Leave-one-study-out cross-validation employed during training.
- PCA to generate features for each gene

Filtering Observations

Studies containing pam50 predictions

- Studies 1, 2, 3, 17 and 18 contained only NA values for pam50
- Data from them are not included for training and testing

```
Study 1: 114 samples, 114 NAs (100.00%), Has pam50 data: FALSE
Study 2: 53 samples, 53 NAs (100.00%), Has pam50 data: FALSE
Study 3: 261 samples, 261 NAs (100.00%), Has pam50 data: FALSE
Study 4: 28 samples, 0 NAs (0.00%), Has pam50 data: TRUE
Study 5: 31 samples, 0 NAs (0.00%), Has pam50 data: TRUE
Study 6: 128 samples, 0 NAs (0.00%), Has pam50 data: TRUE
Study 7: 20 samples, 0 NAs (0.00%), Has pam50 data: TRUE
Study 8: 122 samples, 0 NAs (0.00%), Has pam50 data: TRUE
Study 9: 16 samples, 0 NAs (0.00%), Has pam50 data: TRUE
Study 10: 221 samples, 0 NAs (0.00%), Has pam50 data: TRUE
Study 11: 6 samples, 0 NAs (0.00%), Has pam50 data: TRUE
Study 12: 17 samples, 0 NAs (0.00%), Has pam50 data: TRUE
Study 13: 71 samples, 0 NAs (0.00%), Has pam50 data: TRUE
Study 14: 25 samples, 0 NAs (0.00%), Has pam50 data: TRUE
Study 15: 16 samples, 0 NAs (0.00%), Has pam50 data: TRUE
Study 16: 54 samples, 0 NAs (0.00%), Has pam50 data: TRUE
Study 17: 115 samples, 115 NAs (100.00%), Has pam50 data: FALSE
Study 18: 11 samples, 11 NAs (100.00%), Has pam50 data: FALSE

Studies without pam50 data: 1, 2, 3, 17, 18
Studies with incomplete pam50 data: None
Studies with complete pam50 data: 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16
```

NA Values in remaining studies

- Remove all rows with NA values for pCR
- Assign zero to NA gene expression values

Normalization using biomarker MKI67

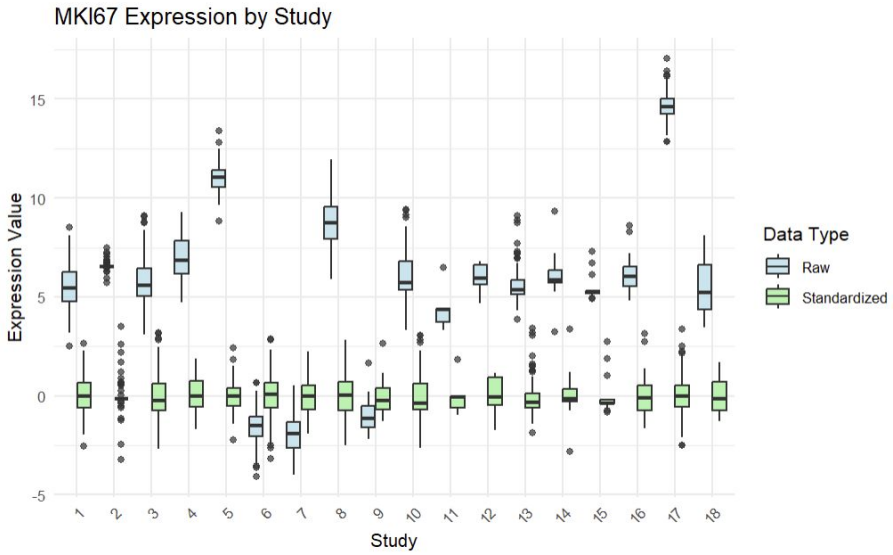
Methodology

- Identify common genes across all datasets
- Filter studies to include only shared genes for consistent analysis

Correct for Batch Effects

- Normalize gene expression data using MKI67 (proliferation marker)
  - Subtracting MKI67 expression
  - Scaling by standard deviation of MKI67 expression in each study
- Minimize differences caused by batch effects
- Preserve Data Structure
- Split data back into individual studies post-normalization
- Verify that dimensions (number of genes and samples) remain consistent

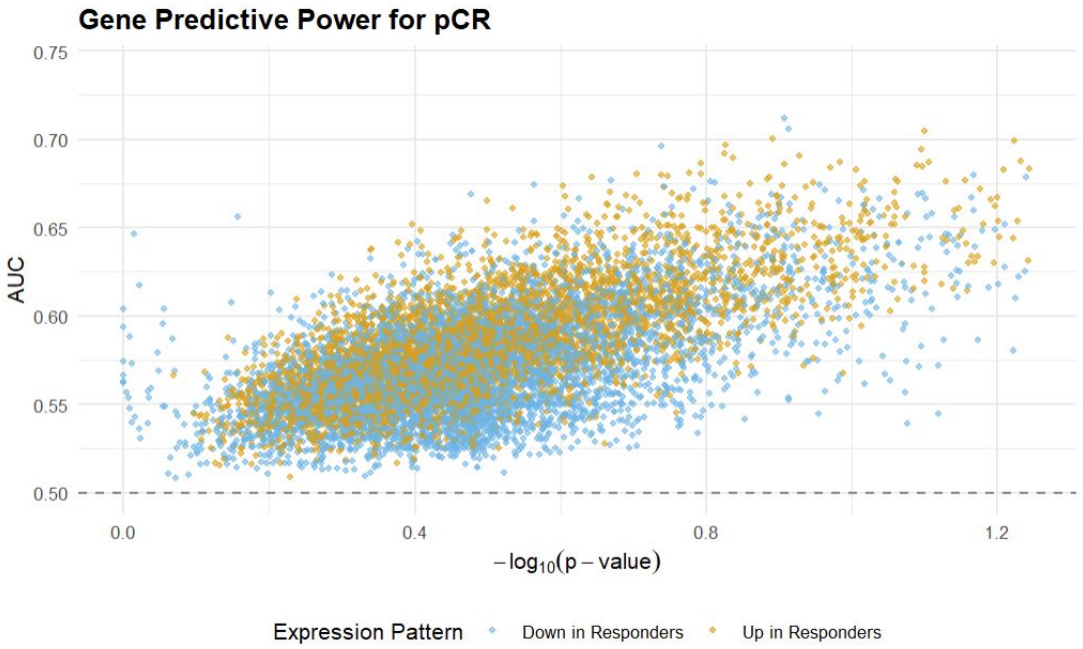
Comparison



Source: Ki-67 labeling index is a predictive marker for a pathological complete response to neoadjuvant chemotherapy in breast cancer

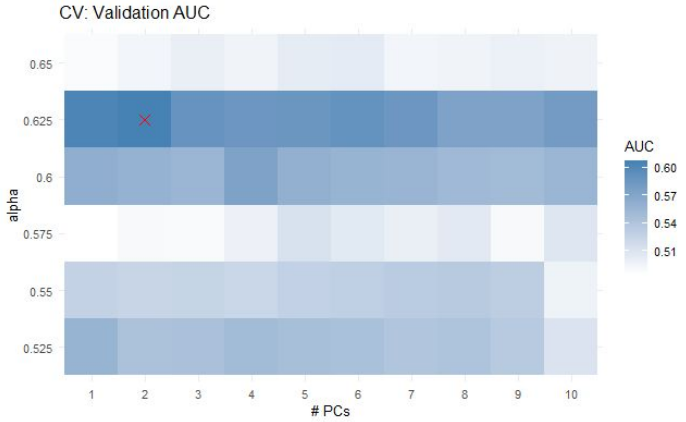
Feature selection

AUC threshold



Using cross-validation to select alpha

```
set.seed(117)
alphas <- seq(0.525, 0.650, 0.025)
kmax <- 10
cat("Searching α ∈ [", min(alphas), "-", max(alphas), "] and k = 1…", kmax, "\n")
cv <- CV_pca(XX_pCR, pCR, alphas, kmax, leave_one = TRUE)

print(cv$heatmap)
cat("→ Best α =", cv$alpha_best, "  k =", cv$k_best, "\n")
```



Feature selection

Clustering based on corr threshold of 0.8

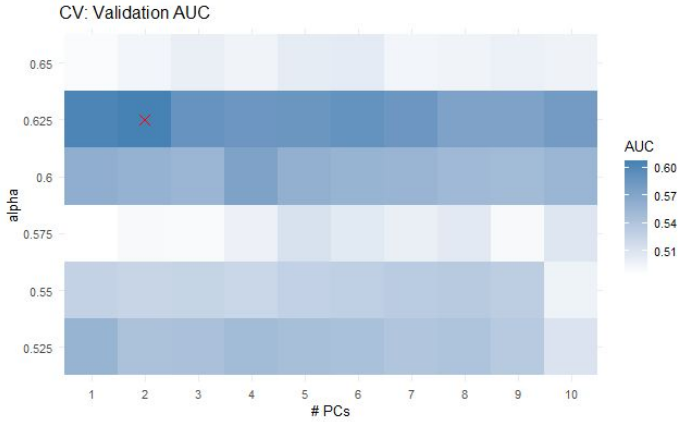
- Assign +1 or -1 weight based on whether AUC is above or below 0.5
- For each cluster, compute the weighted sum of all genes in that cluster
- Using PCA, obtain k features, built from genes that both correlate positively and negatively with pCR

```r
# Function to cluster genes based on correlation
corr_cluster <- function(expr_mat, corr_thr = 0.8) {
  # Calculate the pairwise correlation matrix
  cor_mat <- cor(t(expr_mat), method = "pearson", use = "pairwise.complete.obs")

  # Take absolute values of correlations for clustering purposes
  abs_cor_mat <- abs(cor_mat)

  # Create a distance matrix where distance = 1 - abs(correlation)
  dist_mat <- as.dist(1 - abs_cor_mat)

  # Perform hierarchical clustering using average linkage
  hc <- hclust(dist_mat, method = "average")

  # Cut the dendrogram at a height corresponding to the threshold
  cluster_assignment <- cutree(hc, h = 1 - corr_thr)

  # Organize genes into groups
  groups <- split(names(cluster_assignment), cluster_assignment)

  # Sort groups by size (largest first)
  group_sizes <- sapply(groups, length)
  groups_sorted <- groups[order(group_sizes, decreasing = TRUE)]

  return(groups_sorted)
}
```

Using cross-validation to select k

-

```r
set.seed(117)
alphas <- seq(0.525, 0.650, 0.025)
kmax <- 10
cat("Searching α ∈ [", min(alphas), "-", max(alphas), "] and k = 1…", kmax, "\n")
cv <- CV_pca(XX_pCR, pCR, alphas, kmax, leave_one = TRUE)

print(cv$heatmap)
cat("→ Best α =", cv$alpha_best, "  k =", cv$k_best, "\n")
```



Source: WGCNA: Zhang & Horvath (2005) - "A General Framework for Weighted Gene Co-Expression Network Analysis"

Training and testing data sets

Method

- **Methodology:** Leave-one-study-out
- Chose study 10 because it had the most samples -> model eval of least variance
- Through studies 4 to 16

Benchmarking

- **Goal**: Develop a classifier to predict pCR status
- **Approach**:
  - Train using biomarker expression data (XX_pCR)
  - Predict binary pCR response
  - Compare performance against pam50 benchmark
- **Evaluation**: Use metrics like AUC to assess model performance

Source: WGCNA: Zhang & Horvath (2005) - "A General Framework for Weighted Gene Co-Expression Network Analysis"

Baseline Model: Logistic Regression

Merge all studies together to generate one big XX_pCR dataset including outcome, study ID

```
j <- 10
k_best <- 2

# 1) split indices
idx_tr <- which(YY$study != j)
idx_te <- which(YY$study == j)

# 2) fit logistic model on training subset only
lr_mod <- fit_model(
  predictors = P,
  YY_train   = YY$response[idx_tr],
  indexes    = idx_tr,
  k          = k_best,
  lr         = TRUE
)

# 3) compute AUCs
df_tr <- as.data.frame(P[idx_tr,]); colnames(df_tr) <- paste0("P",1:k_best)
df_te <- as.data.frame(P[idx_te,]); colnames(df_te) <- paste0("P",1:k_best)

lr_train_auc <- auc_safe(
  predict(lr_mod, newdata = df_tr, type="response"),
  YY$response[idx_tr]
)
lr_test_auc <- auc_safe(
  predict(lr_mod, newdata = df_te, type="response"),
  YY$response[idx_te]
)
```

- **Only provides appropriate fits when the true response is a logistic function of the features.**
- train two flexible models (namely, a random forest classifier and an gradient boosting classifier) on top of a baseline logistic regression model
- Flexible because they can (essentially) fit arbitrarily well to a prescribed training set
- Improvement:
  - Consider applying regularization (e.g., L1 or L2) to address multicollinearity among predictors.
  - Validate the model across multiple folds or studies for further robustness.

Old but gold!

Model 2: Random Forest

Limited Depth of Tree & Number of Samples Per Leaf

```
# 1) train/test indices
idx_tr <- which(YY$study != j)
idx_te <- which(YY$study == j)

# 2) pull out and rename your predictors to P1…Pk
df_tr <- as.data.frame(P[idx_tr, , drop = FALSE])
df_te <- as.data.frame(P[idx_te, , drop = FALSE])
colnames(df_tr) <- colnames(df_te) <- paste0("P", seq_len(k_best))

# 3) fit the RF on the training data frame
rf_mod <- randomForest(
  x        = df_tr,
  y        = as.factor(YY$response[idx_tr]),
  ntree    = 500,
  mtry     = floor(sqrt(k_best)),
  maxnodes = 10,          # cap max tree depth
  nodesize = 5            # minimum number of samples per leaf
)

# 4) get predicted probabilities
rf_train_probs <- predict(rf_mod, newdata = df_tr, type = "prob")[,2]
rf_test_probs  <- predict(rf_mod, newdata = df_te, type = "prob")[,2]

# 5) compute AUCs
rf_train_auc <- auc_safe(rf_train_probs, YY$response[idx_tr])
rf_test_auc  <- auc_safe(rf_test_probs,  YY$response[idx_te])
```

- Considered refining feature selection, adjusting hyperparameters
- Balanced Hyperparameters:
    - The choice of mtry, maxnodes, and nodesize is aimed at balancing model complexity and generalization.
- Advantages of Random Forest:
    - Handles high-dimensional data effectively, which is common in gene expression datasets.
- Robust against overfitting, particularly with the use of default bootstrapping and ensemble averaging.
- Suggestions for Further Enhancements:

Model 3: Gradient Boosting Machine

MODERNIZED & EFFECTIVE Classifer

```r
# 1. Extract predictors and response
X_tr <- as.data.frame(P[idx_tr, , drop = FALSE])
X_te <- as.data.frame(P[idx_te, , drop = FALSE])
y_tr <- YY$response[idx_tr]
y_te <- YY$response[idx_te]

# 2. Fit GBM model
gbm_mod <- gbm::gbm(
  formula = y_tr ~ .,
  data = data.frame(y_tr = y_tr, X_tr),
  distribution = "bernoulli",
  n.trees = 100,
  interaction.depth = 3,
  shrinkage = 0.05,
  n.minobsinnode = 5,
  verbose = FALSE
)

# 3. Find best number of trees via internal cross-validation (if using cv.folds)
best_iter <- gbm::gbm.perf(gbm_mod, method = "OOB", plot.it = FALSE)

# 4. Predict
gbm_train_probs <- predict(gbm_mod, newdata = X_tr, n.trees = best_iter, type = "response")
gbm_test_probs  <- predict(gbm_mod, newdata = X_te, n.trees = best_iter, type = "response")
```

- Iteratively combine weak decision trees by sequentially fixing error
- Excels at capturing nonlinearities and high-order interactions without manual feature construction.
- Offers fine-grained control via multiple regularization knobs (learning rate, tree depth, early stopping)
- In practice often outperforms single models like random forests—at the cost of more tuning
- See results…

Interpretation

Overview

- **AUC measurements**
  - Successful strategy: rigorous feature engineering with flexible models.
  - Simple logistic regression on our features > Prosigna's pam50 features.
  - More flexible models improved performance further (GBM)
- GBM showed good ability to generalize
  - Test AUC slightly higher than train AUC

Suggestions

- Suggests need for stronger dimensionality reduction/feature selection with Model III.
- Random forest model achieved highest test AUC of all models but also most overfitting
- Need to benchmark against pam50

Model	Train AUC	Test AUC	Gap
Logistic Regression	0.674	0.754	-0.079
Random Forest	0.830	0.705	0.125
GBM	0.719	0.755	-0.037

Other metrics

AUC

- Everyone uses this
- Compared with concordance index, better visualizations

Brier score

- The Brier score measures the accuracy of probabilistic predictions. It is a proper scoring rule that evaluates both calibration and discrimination, providing a comprehensive measure of model performance
- Possibly dominated by large majority class

Log loss

- Penalizes incorrect predictions and is particularly useful for evaluating models that output probabilities. It provides a measure of the model's confidence in its predictions.
- Possibly use weighting to compare models given imbalanced data

Future Steps

Overview

- **Three Main Extensions**:
  - Use feature importance scores from the GBM model to analyze which predictors most strongly contribute to pCR prediction
  - Consider applying regularization (e.g., L1 or L2) to address multicollinearity among predictors
  - Account for non-linear relationships - for PCA (kernel) and for Pearson correlation