BIOS226 - Topic 5 - Supervised Learning (Part 2)

The Tumor Subtype Classification Pipeline

Dr. Robert Treharne

From Data to Decision: The Pipeline

We will use our synthetic breast cancer gene expression dataset generated in R.

- 120 patients
- 10,000 gene features
- Subtypes: Luminal_A vs Basal_like
- Logistic regression classifier

Goal: Predict tumor subtype from gene expression profiles.

1. Define the Prediction Problem

Clinical question: Can we classify a tumor as *Basal_like* or *Luminal_A* based on gene expression?

Model task: Binary classification.

- Input: 10,000 gene expression values per patient
- Output: Probability tumor is Basal_like

Positive class: **Basal_like**

2. Validate the Data Structure

Before modeling, verify:

- Column structure (`Patient_ID`, `Subtype`, `Gene_1 ... Gene_n`)
- Exactly two classes present
- All gene columns are numeric
- No schema inconsistencies

Why? Garbage in leads to garbage out.

3. Stratified Train/Test Split

Split the data:

- 80% training
- 20% test
- Stratified by subtype

Why stratified? It maintains class balance in both sets.

The test set is held out and **never** used during training.

4. Feature Selection (Train Only)

Rank genes by absolute difference in mean expression between classes.

Select top K genes (e.g., 25).

Important: Feature selection is performed on training data only.

Why? To avoid data leakage.

5. Scaling & Normalisation (Train Only)

Gene expression values vary in magnitude.

Standardise each selected gene:

- Subtract mean
- Divide by standard deviation

Scaling parameters are learned from training data only and then applied to test data.

6. Cross-Validation on Training Data

Perform k-fold cross-validation (e.g., 5-fold):

1. Split training set into 5 parts
2. Train on 4 folds
3. Validate on 1 fold
4. Repeat 5 times

Outputs:

- AUC per fold
- Precision per fold

Purpose: Estimate **model stability** before touching test set.

7. Fit the Logistic Regression Model

Logistic regression estimates:

```
P(Basal_like | gene expression)
```

Model form:

```
log(p / (1 - p)) = beta0 + beta1x1 + beta2x2 + ...
```

Output: Probability between 0 and 1 for each patient.

8. Confusion Matrix: Understanding Errors

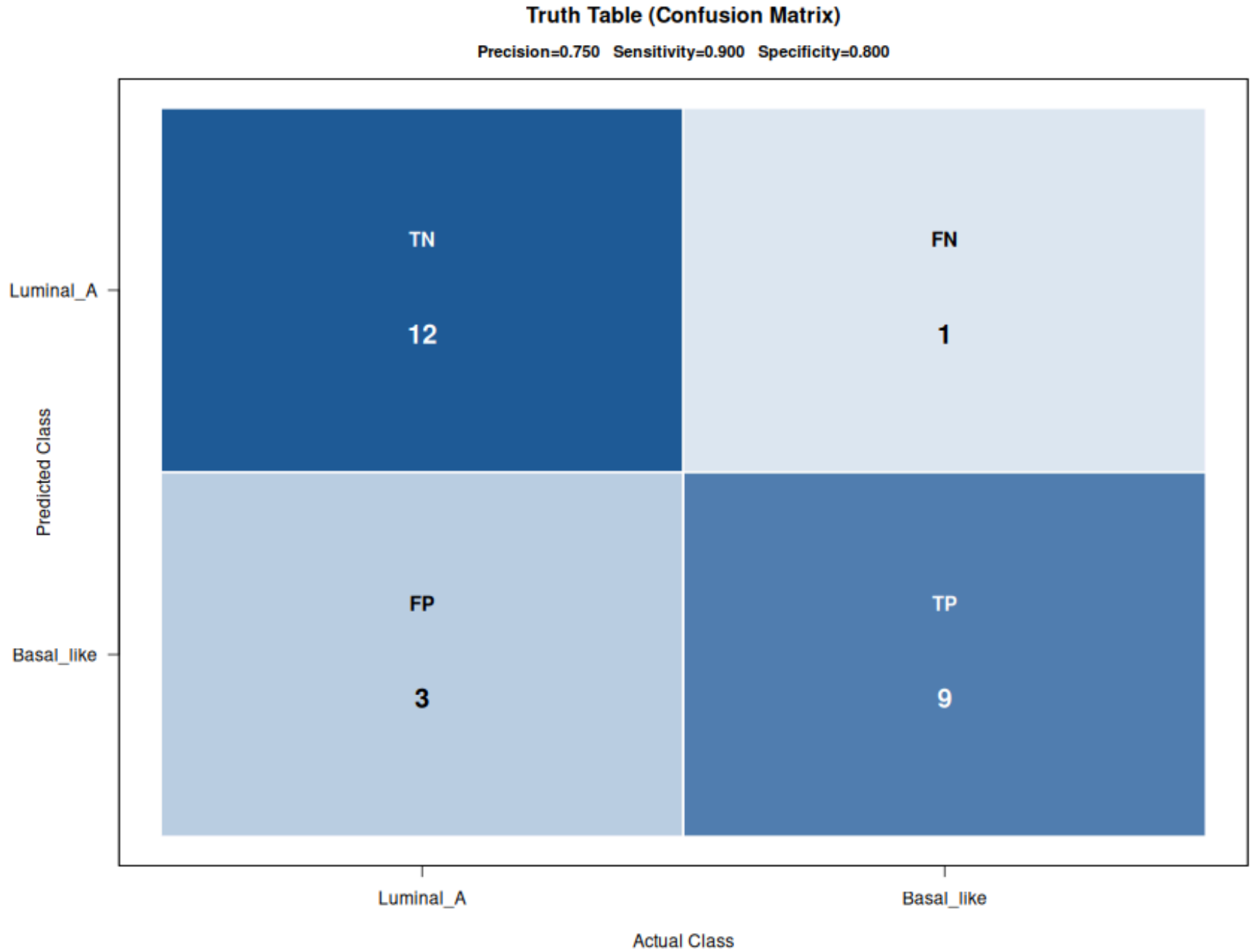
Using chosen threshold (e.g., 0.85 - high specificity! Why?), compute:

- True Positives (TP)
- False Positives (FP)
- True Negatives (TN)
- False Negatives (FN)

From this we calculate:

- Precision
- Sensitivity
- Specificity

This describes performance at one threshold.


Truth Table (Confusion Matrix)
Precision=0.750 Sensitivity=0.900 Specificity=0.800
TN
12
FN
1
FP
3
TP
9
Luminal_A
Basal_like
Predicted Class
Luminal_A
Basal_like
Actual Class

9. ROC Curve & AUC

The model outputs probabilities.

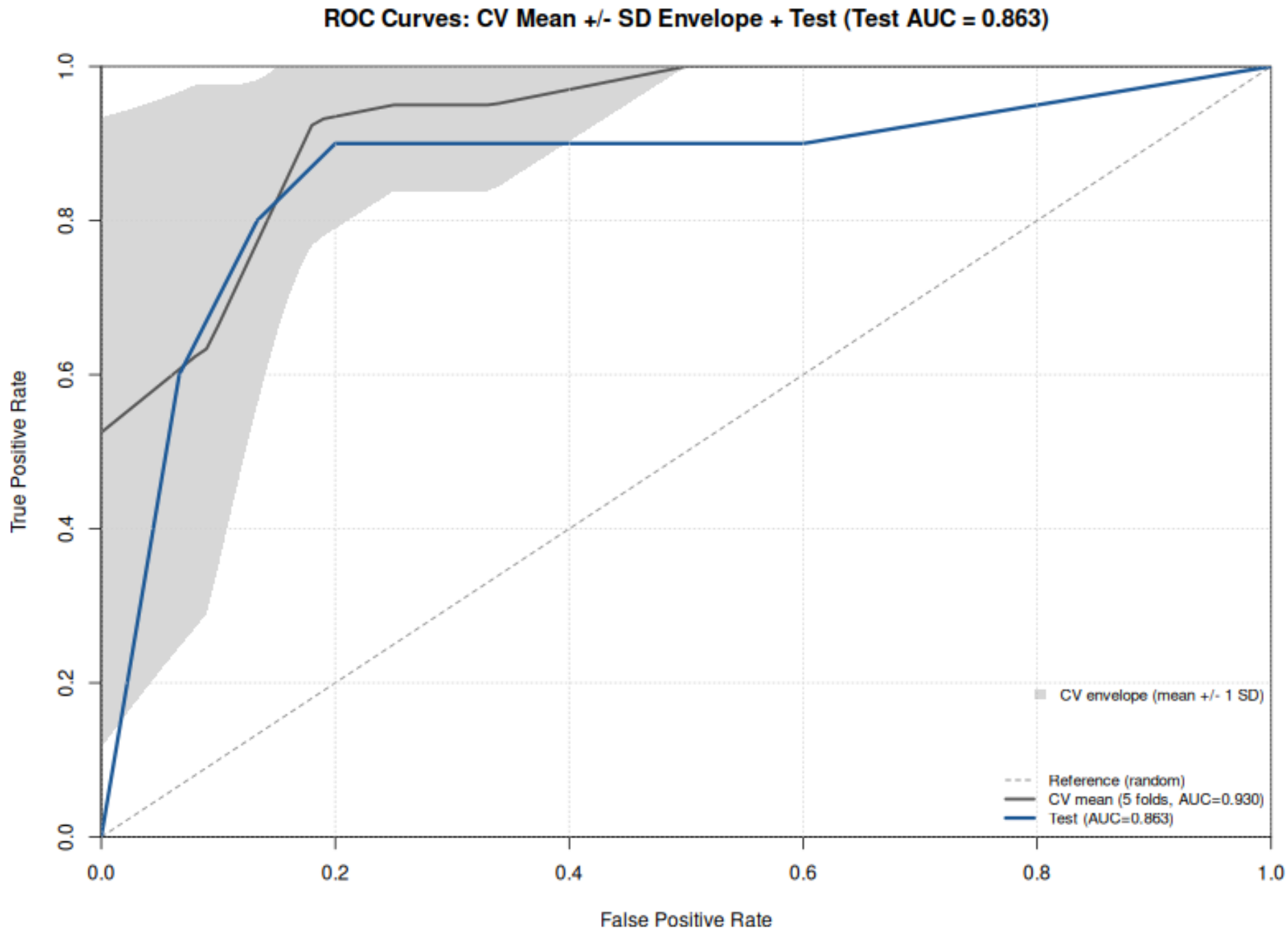
If threshold varies, sensitivity and false positive rate change.

ROC curve shows performance across all thresholds:

- X-axis: False Positive Rate
- Y-axis: True Positive Rate

AUC measures discrimination ability:

- 0.5 = random
- 1.0 = perfect


ROC Curves: CV Mean +/- SD Envelope + Test (Test AUC = 0.863)
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
CV envelope (mean +/- 1 SD)
Reference (random)
CV mean (5 folds, AUC=0.930)
Test (AUC=0.863)

10. Final Test Evaluation and Discipline

After all training steps, evaluate once on the held-out test set.

Report:

- Confusion matrix
- Precision
- ROC AUC
- MSE (probability-based)
- R squared (probability-based)

This gives an unbiased estimate of real-world performance.

Pipeline Summary

1. Define question
2. Validate data
3. Split train/test
4. Select features (train only)
5. Scale data (train only)
6. Cross-validate
7. Fit model
8. Evaluate with confusion matrix
9. Evaluate with ROC/AUC
10. Report final test performance