STA 141C PCA testing BC

Ricky Nunez

2025-05-10

```
library(caret)

# 1. Build a data.frame of **all** features + target
full_df <- data.frame(
  data_encoded[ , setdiff(names(data_encoded), "target")],
  target = data_encoded$target
)

# 2. Upsample the entire data set
set.seed(3)
full_upsampled <- upSample(
  x     = full_df[ , setdiff(names(full_df), "target")],
  y     = full_df$target,
  yname = "target"
)

# Check that all classes now have the same count:
table(full_upsampled$target)
```

```
##
##     basal cell_line       HER luminal_A luminal_B    normal
##        41        41        41        41        41        41
```

```
# 3. Now split *that* balanced set 80/20
set.seed(123)
split_idx <- createDataPartition(full_upsampled$target, p = 0.8, list = FALSE)
train_df <- full_upsampled[ split_idx, ]
test_df  <- full_upsampled[-split_idx, ]

# 4. Extract X/y for both sets
X_train_resampled <- train_df[ , setdiff(names(train_df), "target") ]
y_train_resampled <- train_df$target

X_test_resampled  <- test_df[ , setdiff(names(test_df),  "target") ]
y_test_resampled  <- test_df$target

# 5. (Optional sanity check)
cat("Train rows:", nrow(X_train_resampled), "\n")
```

```
## Train rows: 198
```

```r
cat("Test  rows:", nrow(X_test_resampled), "\n")
```

```
## Test  rows: 48
```

```r
library(ggplot2)
library(dplyr)

# Count per class
type_counts <- data %>%
  count(type) %>%
  mutate(Percentage = n / sum(n) * 100)

# Bar plot of class counts
bar_plot <- ggplot(type_counts, aes(x = type, y = n, fill = type)) +
  geom_bar(stat = "identity") +
  labs(title = "Sample Count by Class", x = "Cancer Subtype", y = "Count") +
  theme_minimal() +
  theme(legend.position = "none")

# Pie chart of class proportions
pie_plot <- ggplot(type_counts, aes(x = "", y = Percentage, fill = type)) +
  geom_bar(stat = "identity", width = 1) +
  coord_polar("y") +
  labs(title = "Class Distribution (Pie Chart)") +
  theme_void() +
  geom_text(aes(label = paste0(round(Percentage, 1), "%")),
            position = position_stack(vjust = 0.5), color = "white", size = 4)

# Combine the two plots using gridExtra
library(gridExtra)
```

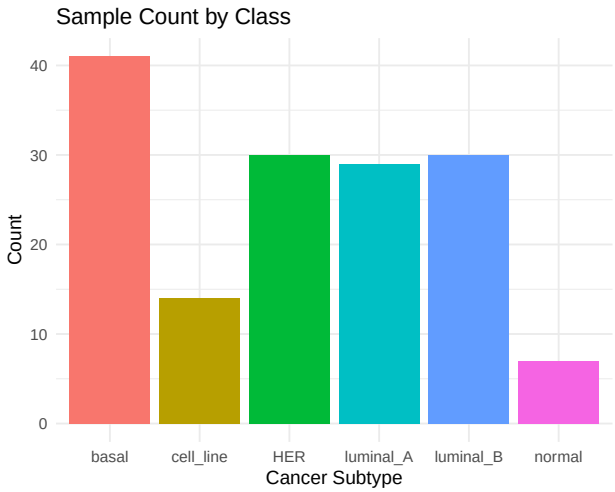
```
## Warning: package 'gridExtra' was built under R version 4.4.2
```

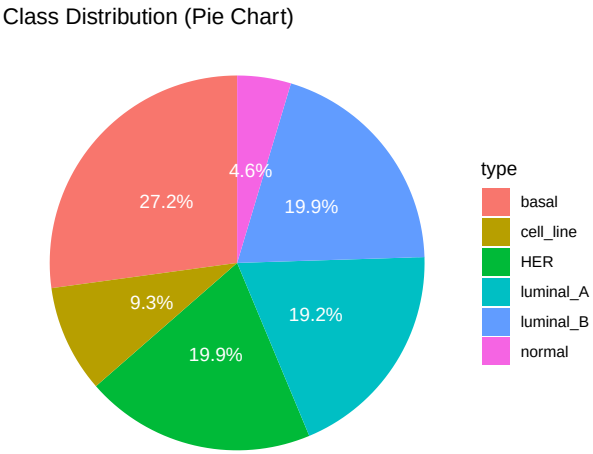
```
##
## Attaching package: 'gridExtra'
```

```
## The following object is masked from 'package:randomForest':
##
##     combine
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```r
grid.arrange(bar_plot, pie_plot, ncol = 2)
```



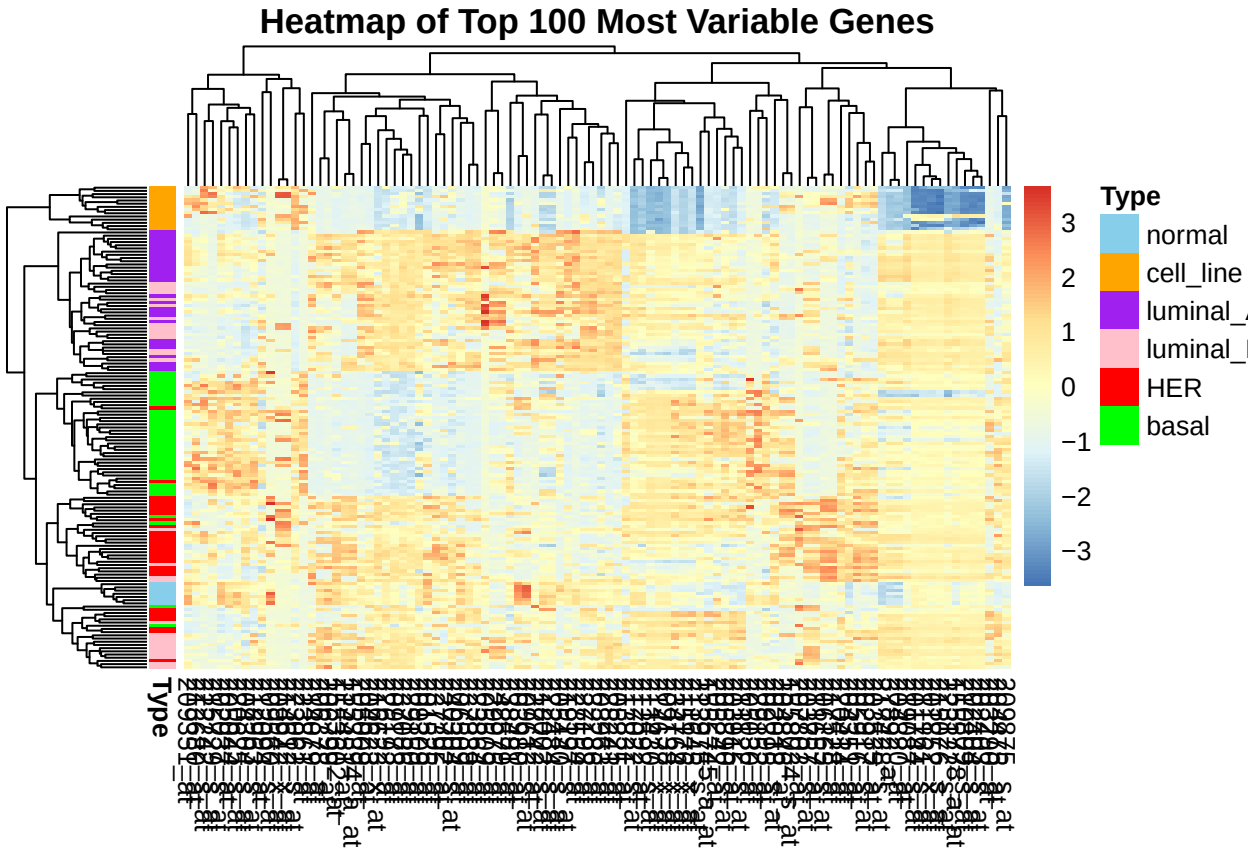



```r
# 1. Extract features (numeric genes only)
features <- data[, !(names(data) %in% c("samples", "type"))]

# 2. Calculate variance of each gene
gene_vars <- apply(features, 2, var)

# 3. Select top 100 most variable genes
top_genes <- names(sort(gene_vars, decreasing = TRUE)[1:100])

# 4. Subset features to just those top genes
heatmap_data <- features[, top_genes]

# 5. Scale each gene (Z-score per column)
heatmap_scaled <- scale(heatmap_data)

# 6. Set rownames to sample names (if missing, create dummy rownames)
if (is.null(rownames(heatmap_scaled))) {
  rownames(heatmap_scaled) <- paste0("Sample", 1:nrow(heatmap_scaled))
}

data$type <- factor(data$type, levels = c("normal", "cell_line", "luminal_A", "luminal_B", "HER", "basal

# 7. Create annotation frame (sample type labels)
annotation_df <- data.frame(Type = data$type)
rownames(annotation_df) <- rownames(heatmap_scaled)

# 8. Define annotation colors
annotation_colors <- list(
  Type = c(
    "normal" = "skyblue",
    "cell_line" = "orange",
    "luminal_A" = "purple",
    "luminal_B" = "pink",
    "HER" = "red",
    "basal" = "green"
  )
)
```

```r
# 9. Load library and plot
library(pheatmap)
```

```
## Warning: package 'pheatmap' was built under R version 4.4.3
```

```r
pheatmap(
  mat = heatmap_scaled,
  annotation_row = annotation_df,
  annotation_colors = annotation_colors,
  cluster_rows = TRUE,
  cluster_cols = TRUE,
  show_rownames = FALSE,
  main = "Heatmap of Top 100 Most Variable Genes"
)
```




```r
# ------- Scale resampled train/test -----------
# X_train_resampled and X_test_resampled already exist
# 1. Scale training predictors
X_train_scaled <- scale(X_train_resampled)
# 2. Save the centering & scaling attributes
train_center <- attr(X_train_scaled, "scaled:center")
train_scale  <- attr(X_train_scaled, "scaled:scale")

# 3. Apply the same transform to test predictors
```

```r
X_test_scaled <- scale(
  X_test_resampled,
  center = train_center,
  scale  = train_scale
)

# 4. Turn them into data.frames (if you want)
X_train_scaled <- as.data.frame(X_train_scaled)
X_test_scaled  <- as.data.frame(X_test_scaled)

# Quick QC
cat("Train means (should be ~0):", round(colMeans(X_train_scaled)[1:5], 3), "\n")
```

```
## Train means (should be ~0): 0 0 0 0 0
```

```r
cat("Test means (not exactly 0):", round(colMeans(X_test_scaled)[1:5], 3), "\n")
```

```
## Test means (not exactly 0): 0.016 -0.094 -0.013 0.148 -0.075
```

```r
library(ggplot2)
library(dplyr)

# 1. Count class frequencies per set (using the resampled labels)
train_df <- data.frame(Set = "Train", Label = y_train_resampled)
test_df  <- data.frame(Set = "Test",  Label = y_test_resampled)
combined_df <- bind_rows(train_df, test_df)

# 2. Create histogram-style bar plots for each set
ggplot(combined_df, aes(x = Label)) +
  geom_bar(fill = "#3E92CC") +
  facet_wrap(~ Set, nrow = 1, scales = "free_y") +
  labs(
    title = "Label Distribution After Upsampling: Train vs Test",
    x = "Class Label",
    y = "Count"
  ) +
  theme_minimal() +
  theme(strip.text = element_text(size = 14))
```


```r
#Run PCA on the SCALED, RESAMPLED training set
pca_result <- prcomp(
  X_train_scaled,   # scaled & resampled train features
  center = TRUE,    # these are redundant when you already scaled, but safe
  scale. = TRUE
)

# Capture how much variance each PC explains
explained_var <- pca_result$sdev^2
prop_var      <- explained_var / sum(explained_var)
cum_var       <- cumsum(prop_var)
num_pc_95     <- which(cum_var >= 0.95)[1]
cat("Number of PCs to capture 95% variance:", num_pc_95, "\n")
```

```
## Number of PCs to capture 95% variance: 100
```

```r
# Build PCA train & test matrices
# Training scores:
X_pca_train <- as.data.frame(pca_result$x[, 1:num_pc_95])

# Scale your test set exactly the same way *before* projecting:
test_scaled <- scale(
  X_test_scaled,                  # your already scaled test data
  center = pca_result$center,      # prcomp's internal center (mean)
  scale  = pca_result$scale        # prcomp's internal scale (sd)
)

# Project onto the first num_pc_95 PCs:
X_pca_test_manual <- test_scaled %*% pca_result$rotation[, 1:num_pc_95]

# Turn into data frame and name columns:
df_pca_test <- as.data.frame(X_pca_test_manual)
colnames(df_pca_test) <- paste0("PC", 1:ncol(df_pca_test))

# Attach the resampled test labels:
df_pca_test$target <- y_test_resampled
```

```
################################

################################
# Now df_pca_test is a ready-to-use test set with PCA features + label
str(df_pca_test)
```

```
## 'data.frame':    48 obs. of  101 variables:
##  $ PC1   : num  -115.1 -112.3 -83.9 -32.6 -67.2 ...
##  $ PC2   : num  25.852 -21.226 0.992 11.577 -4.111 ...
##  $ PC3   : num  -9.18 -20.42 -29.82 7.2 40.67 ...
##  $ PC4   : num  89.6 87 55.5 78.2 81.4 ...
##  $ PC5   : num  -7.9 -3.9 25.89 -6.28 24.76 ...
##  $ PC6   : num  60.6 39.8 60.9 -73.2 -28.2 ...
##  $ PC7   : num  -18.82 -15.54 -1.45 40.83 -13.77 ...
##  $ PC8   : num  -10.61 7.11 17.72 -18.88 20.87 ...
##  $ PC9   : num  -3.39 -13.71 14.6 1.9 -4.61 ...
##  $ PC10  : num  30.38 12.74 10.27 37.56 -2.59 ...
##  $ PC11  : num  -7.83 -3.18 -3.41 13.34 -9.81 ...
##  $ PC12  : num  34.28 40.63 1.34 19.92 26.69 ...
##  $ PC13  : num  6.264 0.856 12.916 -10.661 -6.534 ...
##  $ PC14  : num  12.68 14.69 16.83 3.45 -2.37 ...
##  $ PC15  : num  -16.86 -18.01 -22.65 -2.48 -14.03 ...
##  $ PC16  : num  8.76 5.63 2.18 3.16 20.01 ...
##  $ PC17  : num  -18.68 -18.49 -9.32 -1.2 -8.26 ...
##  $ PC18  : num  0.51 9.59 -25.01 9.42 -3.79 ...
##  $ PC19  : num  -7.42 3.22 -23.84 24.74 4.39 ...
##  $ PC20  : num  22.14 13.41 4.08 -13.42 -14.11 ...
##  $ PC21  : num  -1.81 -9.52 11.13 20.85 -10.5 ...
##  $ PC22  : num  -8.51 1.16 -3.5 -10.44 2.07 ...
##  $ PC23  : num  -11.031 -0.815 1.519 -17.886 2.391 ...
##  $ PC24  : num  -4.5 -16.73 -13.94 0.38 10.02 ...
##  $ PC25  : num  -1.17 11.43 17.97 -2.66 4.04 ...
##  $ PC26  : num  -13.41 -10.53 -19.23 3.73 3.02 ...
##  $ PC27  : num  12.766 -5.656 10.146 11.436 0.706 ...
##  $ PC28  : num  5.58 -2.03 15.55 -2.3 -3.04 ...
##  $ PC29  : num  3.887 0.996 -5.955 13.4 2.641 ...
##  $ PC30  : num  5.21 1.26 7.32 7.84 3.38 ...
##  $ PC31  : num  10.35 -2.07 15.42 -4.49 -14.5 ...
##  $ PC32  : num  -8.91 13.37 -1.86 -21.45 -6.37 ...
##  $ PC33  : num  -1.37 2.14 -6.15 7.06 7.86 ...
##  $ PC34  : num  3.623 0.645 2.002 2.687 8.699 ...
##  $ PC35  : num  1.11 -12.02 10.13 -1.17 3.13 ...
##  $ PC36  : num  -8.353 10.868 10.745 -4.303 -0.365 ...
##  $ PC37  : num  -0.457 -0.421 1.553 12.223 3.628 ...
##  $ PC38  : num  -5.477 7.779 -1.273 0.786 2.545 ...
##  $ PC39  : num  11.01 -6.89 -6.91 8.85 -9.31 ...
##  $ PC40  : num  9.64 9.55 22.7 -3.72 -4.07 ...
##  $ PC41  : num  -9.61 2.66 16.21 -3.53 7.37 ...
##  $ PC42  : num  -9.032 9.021 -13.414 -6.684 0.664 ...
##  $ PC43  : num  -3.45 1.83 -10.52 -2 7.13 ...
##  $ PC44  : num  -14.17 16.66 20.45 6.64 3.46 ...
##  $ PC45  : num  2.54 -3.64 3.17 13.07 14.73 ...
##  $ PC46  : num  -8.55 6.69 -8.43 -9.52 -3.84 ...
```

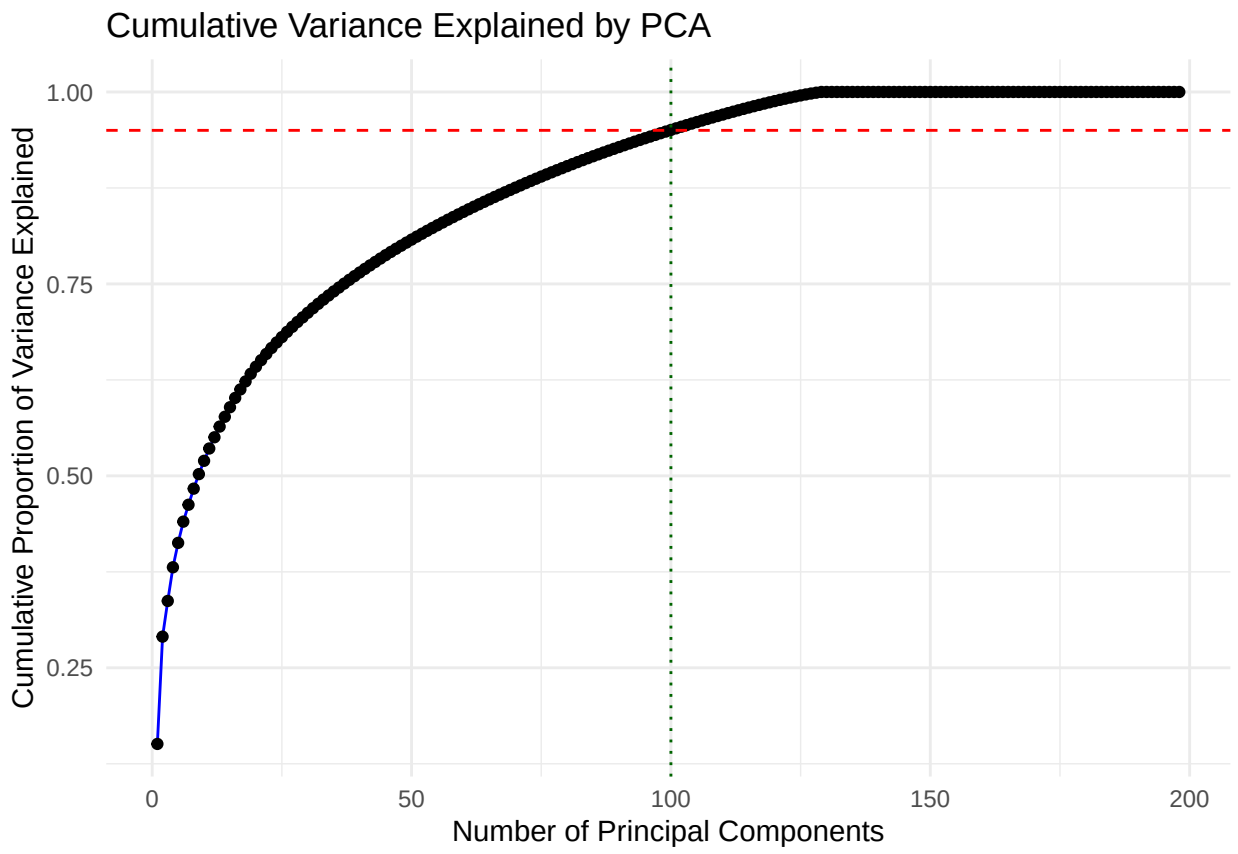
```
##  $ PC47  : num  7.68 -7.22 13.04 -2.52 -1.04 ...
##  $ PC48  : num  -4.53 2.17 -3.23 5.71 5.78 ...
##  $ PC49  : num  -4.64 3.9 2.15 12.18 -4.41 ...
##  $ PC50  : num  -3.32 -9.8 4.77 -3.41 -4.62 ...
##  $ PC51  : num  -2.7615 -0.0313 4.4347 -1.1308 4.9966 ...
##  $ PC52  : num  -2.06 6.25 1.8 1.52 -2.27 ...
##  $ PC53  : num  -2.0785 0.0973 -2.3552 1.0514 -0.283 ...
##  $ PC54  : num  -3.851 10.116 0.475 -19.352 -0.825 ...
##  $ PC55  : num  1.22 -4.06 14.27 1.26 1.49 ...
##  $ PC56  : num  0.0203 6.0283 3.1705 -3.7706 -5.9468 ...
##  $ PC57  : num  0.225 5.248 -8.573 -0.357 1.525 ...
##  $ PC58  : num  -2.13 -9.61 5.83 4.19 -5.6 ...
##  $ PC59  : num  4.25 2.14 -9.31 -1.57 2.59 ...
##  $ PC60  : num  -4.56 8.93 12.58 -4.42 4.96 ...
##  $ PC61  : num  4 5.57 -5.34 0.55 2.64 ...
##  $ PC62  : num  1.3 10.85 -5.85 -2.29 -2.6 ...
##  $ PC63  : num  -1 7.18 9.45 -1.45 -2.15 ...
##  $ PC64  : num  16.8326 0.0879 -2.4441 11.1786 3.4491 ...
##  $ PC65  : num  2.1 -2.67 -6.11 2.72 5.29 ...
##  $ PC66  : num  2.76 -7.08 2.82 -1.83 -1.34 ...
##  $ PC67  : num  9.601 -3.285 -0.546 -0.524 -8.703 ...
##  $ PC68  : num  1.784 -1.632 0.828 5.518 -10.72 ...
##  $ PC69  : num  -6.87 6.66 4.06 -7.4 -7.54 ...
##  $ PC70  : num  -3.26 9.992 -7.001 -0.703 6.31 ...
##  $ PC71  : num  -1.67 7.87 7.03 -5.84 4.23 ...
##  $ PC72  : num  -2.92 3.14 -4.36 5.57 -1.49 ...
##  $ PC73  : num  9.04 -10.86 -2.36 7.78 -2.89 ...
##  $ PC74  : num  10.62 6.47 -5.11 5.65 -3.15 ...
##  $ PC75  : num  1.16 8.16 1.84 2.22 -2.13 ...
##  $ PC76  : num  -0.618 -3.604 7.762 -3.636 2.677 ...
##  $ PC77  : num  -5.22 -2.15 9.47 1.12 -6.23 ...
##  $ PC78  : num  0.446 4.137 3.422 0.873 0.466 ...
##  $ PC79  : num  -10.769 -4.181 8.679 6.565 0.438 ...
##  $ PC80  : num  1.44 3.56 -5.15 -1.38 3.89 ...
##  $ PC81  : num  -5.85 5.68 -1.2 -2.54 6.24 ...
##  $ PC82  : num  -4.243 -10.772 -9.578 -0.915 0.931 ...
##  $ PC83  : num  -2.302 -2.151 0.905 -0.932 -1.916 ...
##  $ PC84  : num  -1.01 -2.34 -2.2 1.21 4.76 ...
##  $ PC85  : num  1.14 1.38 7.2 3.28 -3.47 ...
##  $ PC86  : num  2.36 1.392 -2.472 0.173 4.283 ...
##  $ PC87  : num  3.784 7.42 -0.943 6.217 -0.788 ...
##  $ PC88  : num  4.982 -2.144 4.27 2.599 0.505 ...
##  $ PC89  : num  -4.88 6.27 4.73 6.66 0.18 ...
##  $ PC90  : num  -2.81 8.12 -2.36 5.77 9.82 ...
##  $ PC91  : num  1.137 2.589 3.362 -0.684 9.109 ...
##  $ PC92  : num  0.843 -6.332 -1.552 9.071 -1.847 ...
##  $ PC93  : num  -1.308 -1.351 -8.215 -0.59 -0.816 ...
##  $ PC94  : num  -3.13 2.47 2.12 5.09 -3.39 ...
##  $ PC95  : num  -5 -3 -2.02 2.76 -1.34 ...
##  $ PC96  : num  -0.528 -1.294 5.87 2.667 -1.678 ...
##  $ PC97  : num  -1.64 -6.67 -3.27 7.02 3.65 ...
##  $ PC98  : num  2.86 -4.94 -14.66 4.25 -6.73 ...
##  $ PC99  : num  -1.432 3.768 0.312 -1.003 7.477 ...
##   [list output truncated]
```

```r
head(df_pca_test)
```

```
##           PC1         PC2        PC3      PC4       PC5       PC6        PC7
## 1  -115.07810  25.8520939  -9.182312 89.60515 -7.898436  60.64256 -18.823366
## 2  -112.30116 -21.2258247 -20.419675 86.97041 -3.897780  39.84476 -15.540956
## 6   -83.92638   0.9915527 -29.816114 55.53777 25.888927  60.88323  -1.453388
## 16  -32.57436  11.5767674   7.203085 78.24862 -6.276079 -73.24335  40.826825
## 18  -67.18073  -4.1111855  40.669375 81.36324 24.760352 -28.16178 -13.769588
## 20  -77.75749 -56.2590646  17.169653 85.71806 35.366901 -22.98550  16.489293
##           PC8        PC9     PC10      PC11      PC12        PC13      PC14
## 1  -10.613859  -3.393865 30.38258 -7.826666 34.281812   6.2641885 12.679071
## 2    7.108108 -13.709135 12.74043 -3.180350 40.631613   0.8561359 14.694746
## 6   17.715031  14.597813 10.26886 -3.405729  1.340348  12.9157665 16.826937
## 16 -18.879088   1.896261 37.55790 13.338263 19.917930 -10.6605644  3.451226
## 18  20.867546  -4.609878 -2.59227 -9.809659 26.693881  -6.5341663 -2.372352
## 20 -16.919456 -25.571173 41.77423  8.235422 11.058152 -13.7888295 -4.771512
##          PC15      PC16       PC17        PC18       PC19       PC20       PC21
## 1  -16.855994  8.761610 -18.676749   0.5102544  -7.416843  22.141462  -1.813783
## 2  -18.011661  5.631409 -18.491698   9.5872101   3.224146  13.411687  -9.519599
## 6  -22.647493  2.176561  -9.315165 -25.0091577 -23.837526   4.076281  11.126531
## 16  -2.481905  3.159492  -1.202060   9.4225424  24.739928 -13.422591  20.849526
## 18 -14.032948 20.013084  -8.256896  -3.7886971   4.387107 -14.108048 -10.502031
## 20   5.285176 21.626277  -2.316350  -5.5246440  20.425357 -22.744882   6.136169
##          PC22        PC23        PC24      PC25       PC26       PC27      PC28
## 1   -8.505009 -11.0309624  -4.5028770 -1.170202 -13.413794 12.7657899  5.579213
## 2    1.161332  -0.8147475 -16.7257534 11.426668 -10.528762 -5.6560528 -2.025806
## 6   -3.496819   1.5185744 -13.9402439 17.968815 -19.232529 10.1456168 15.552451
## 16 -10.438105 -17.8863545   0.3795204 -2.661070   3.730447 11.4360547 -2.298865
## 18   2.066718   2.3906942  10.0226087  4.039480   3.022211  0.7061623 -3.040759
## 20  -5.226814   5.4110556   3.2647064  2.057033  -6.464778 -0.4763302 -3.478591
##          PC29      PC30        PC31       PC32      PC33      PC34       PC35
## 1   3.8867265  5.207589  10.3480738  -8.909540 -1.373585 3.6226217   1.114361
## 2   0.9961141  1.257154  -2.0670798  13.367633  2.142652 0.6453609 -12.017453
## 6  -5.9552339  7.320039  15.4210005  -1.859595 -6.153730 2.0024898  10.128872
## 16 13.4002665  7.836778  -4.4939783 -21.450164  7.059060 2.6867076  -1.173467
## 18  2.6406011  3.375269 -14.5000929  -6.370926  7.857384 8.6988820   3.134439
## 20 -1.4967051 -4.455651   0.4791295 -13.594798 11.556003 5.8628821 -21.768953
##          PC36       PC37       PC38      PC39      PC40       PC41        PC42
## 1  -8.3525618 -0.4566366 -5.4770711 11.005091  9.640886 -9.6143754  -9.0324567
## 2  10.8679135 -0.4205279  7.7785959 -6.892121  9.551439  2.6618729   9.0212774
## 6  10.7450366  1.5530945 -1.2726218 -6.914584 22.696112 16.2069719 -13.4140297
## 16 -4.3032196 12.2229973  0.7861004  8.848221 -3.719621 -3.5348627  -6.6838394
## 18 -0.3649415  3.6281677  2.5454927 -9.305037 -4.072256  7.3650210   0.6638292
## 20 -8.7692742  9.8035488  3.3644778 -7.433235  1.153066 -0.7388553  -5.6832729
##          PC43       PC44      PC45        PC46      PC47      PC48      PC49
## 1   -3.448942 -14.169862  2.540529 -8.55251436  7.677325 -4.533266 -4.635284
## 2    1.826852  16.660031 -3.643143  6.68708805 -7.222383  2.174101  3.901388
## 6  -10.516742  20.453566  3.167604 -8.43490034 13.036260 -3.231114  2.151189
## 16  -1.996343   6.640869 13.067877 -9.52072365 -2.516264  5.712918 12.178423
## 18   7.130453   3.464789 14.734002 -3.83816306 -1.042136  5.783324 -4.412738
## 20  16.542373   6.695437 11.996022  0.06210855 -3.545115  2.337264  1.363538
##         PC50       PC51       PC52        PC53        PC54       PC55
## 1  -3.316499 -2.7615376  -2.061280 -2.07852234  -3.8507298   1.218151
```

```
## 2  -9.796444 -0.0313206   6.246288  0.09729038  10.1155047  -4.056979
## 6   4.771475  4.4347308   1.795170 -2.35523777   0.4754836  14.274400
## 16 -3.410367 -1.1307679   1.521454  1.05137304 -19.3523038   1.260015
## 18 -4.624749  4.9965512  -2.270090 -0.28303823  -0.8251129   1.490323
## 20 -6.428169  5.3795180 -10.993345  6.34407727  -6.8541471 -11.987110
##           PC56       PC57       PC58       PC59       PC60       PC61       PC62
## 1   0.02033082  0.2252367  -2.133630  4.253589 -4.556484  4.0006900  1.302290
## 2   6.02834385  5.2479582  -9.607090  2.136028  8.934976  5.5677987 10.846022
## 6   3.17050988 -8.5731896   5.833600 -9.310759 12.578634 -5.3379992 -5.845306
## 16 -3.77055213 -0.3565777   4.194840 -1.574181 -4.415920  0.5495899 -2.292033
## 18 -5.94675417  1.5246948  -5.595626  2.586351  4.964709  2.6350171 -2.597100
## 20 -2.36917832  1.5163253 -12.464424  9.097611 -1.376799  1.8325531 -1.684840
##         PC63       PC64       PC65       PC66       PC67        PC68       PC69
## 1  -1.004611 16.8326081  2.1048707  2.755867  9.6010914   1.7838206 -6.871493
## 2   7.178389  0.0878787 -2.6700451 -7.076869 -3.2853434  -1.6318404  6.659706
## 6   9.445967 -2.4441002 -6.1075977  2.817078 -0.5464155   0.8275359  4.062723
## 16 -1.447655 11.1786009  2.7247966 -1.825281 -0.5243013   5.5175107 -7.402037
## 18 -2.151592  3.4491491  5.2856368 -1.344248 -8.7027230 -10.7199374 -7.544525
## 20 -6.157472  8.7054108  0.9356359 13.136804 -0.3506886 -10.9241716 -9.364953
##          PC70       PC71       PC72       PC73       PC74       PC75       PC76
## 1  -3.2601057 -1.670005 -2.915941   9.044904 10.622388  1.164374 -0.6184756
## 2   9.9924601  7.873814  3.135773 -10.860000  6.468803  8.160118 -3.6035282
## 6  -7.0005793  7.027097 -4.360718  -2.357238 -5.113794  1.836762  7.7615558
## 16 -0.7027338 -5.837702  5.571868   7.783692  5.650729  2.218543 -3.6362311
## 18  6.3095476  4.227992 -1.493128  -2.885456 -3.145563 -2.132063  2.6771414
## 20  9.3948272  4.304038  4.634322  -0.909436  9.021460 -1.201553  2.1713017
##         PC77       PC78        PC79       PC80       PC81        PC82
## 1  -5.218136  0.4462401 -10.7691746  1.4389490 -5.8480428  -4.2429028
## 2  -2.146099  4.1367462  -4.1809061  3.5587936  5.6824861 -10.7721902
## 6   9.471168  3.4220609   8.6793136 -5.1532089 -1.2036588  -9.5777446
## 16  1.121699  0.8725874   6.5645711 -1.3810738 -2.5432818  -0.9152383
## 18 -6.232903  0.4661867   0.4375346  3.8945409  6.2376987   0.9307161
## 20 -3.458472 -2.8999315   0.3331757  0.8923015 -0.3797354   1.3289301
##          PC83       PC84       PC85       PC86       PC87       PC88       PC89
## 1  -2.3019181 -1.012696  1.137631  2.3601899  3.7840507  4.9816773 -4.8805471
## 2  -2.1512057 -2.337575  1.375214  1.3917060  7.4201518 -2.1439997  6.2689854
## 6   0.9052175 -2.201554  7.200806 -2.4719541 -0.9430233  4.2698666  4.7308929
## 16 -0.9321537  1.206538  3.284698  0.1728186  6.2170345  2.5992761  6.6612501
## 18 -1.9164349  4.763912 -3.466624  4.2831328 -0.7883977  0.5053549  0.1796817
## 20  2.6715145 -4.967480  6.543294  2.1486052  6.1464997 -8.2566979 -5.0279101
##         PC90       PC91       PC92       PC93       PC94       PC95       PC96
## 1  -2.806861  1.1372396  0.8430019 -1.3081864 -3.130950 -4.996208 -0.5275758
## 2   8.116685  2.5887090 -6.3316310 -1.3505787  2.468727 -3.003585 -1.2941184
## 6  -2.359311  3.3616894 -1.5517898 -8.2152091  2.116267 -2.021879  5.8703193
## 16  5.770621 -0.6841807  9.0710043 -0.5904070  5.086117  2.763740  2.6668119
## 18  9.816291  9.1093901 -1.8466078 -0.8162680 -3.386799 -1.341398 -1.6779308
## 20 12.504651  2.9856370 -0.3516973  0.9232087  2.677008  2.676536 -1.4169982
##          PC97       PC98       PC99      PC100 target
## 1  -1.6437585   2.863136 -1.4321330  0.1745717  basal
## 2  -6.6731545  -4.940781  3.7679659  0.7384961  basal
## 6  -3.2719979 -14.664493  0.3120569 -1.2649332  basal
## 16  7.0170397   4.249074 -1.0031300 -3.7941843  basal
## 18  3.6487523  -6.731233  7.4769452  6.0772270  basal
## 20  0.6061547  -3.173150 -1.5165510  1.2800812  basal
```

```r
# Step 4: Create cumulative variance plot
scree_data <- data.frame(
  PC = 1:length(prop_var),
  CumulativeVariance = cum_var
)

ggplot(scree_data, aes(x = PC, y = CumulativeVariance)) +
  geom_line(color = "blue") +
  geom_point() +
  geom_hline(yintercept = 0.95, linetype = "dashed", color = "red") +
  geom_vline(xintercept = num_pc_95, linetype = "dotted", color = "darkgreen") +
  theme_minimal() +
  labs(
    title = "Cumulative Variance Explained by PCA",
    x = "Number of Principal Components",
    y = "Cumulative Proportion of Variance Explained"
  )
```

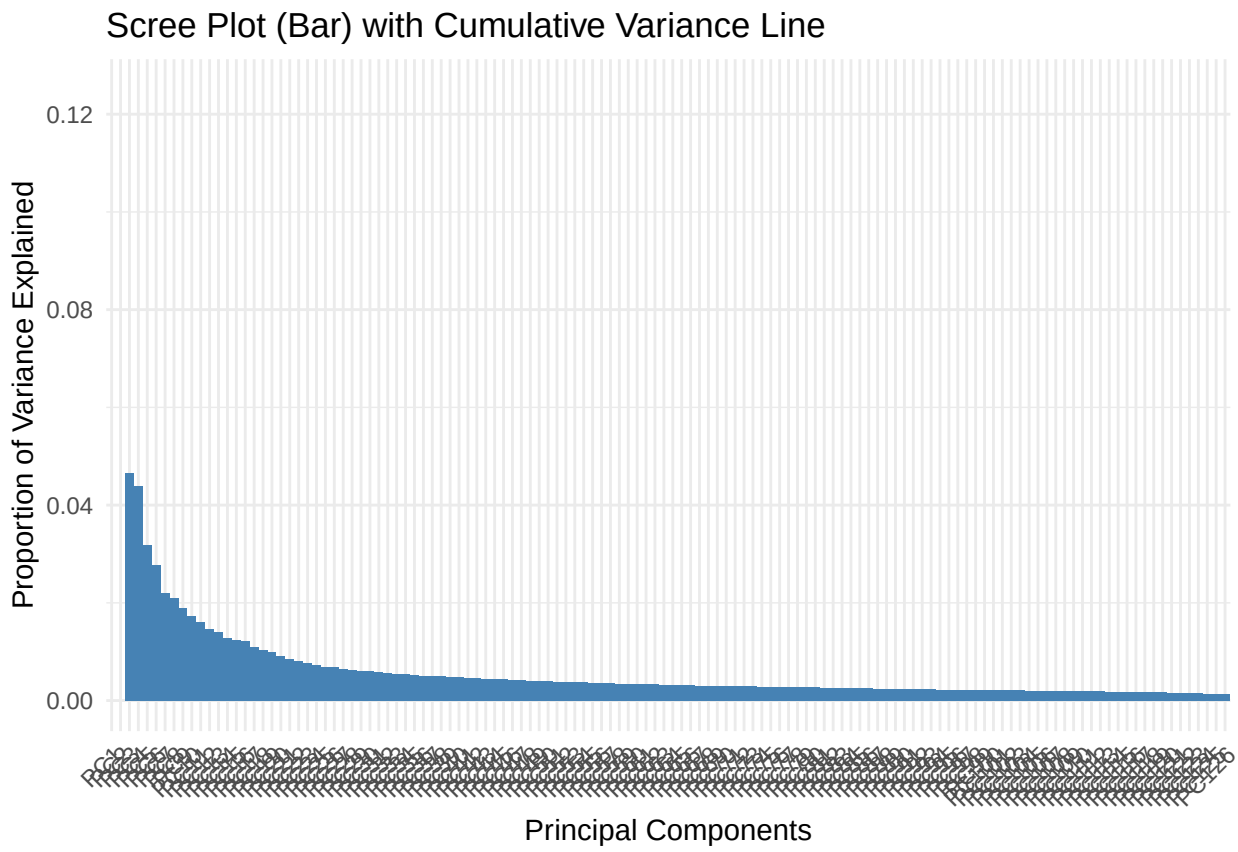



```r
############################bar plot
# Step 1: Variance calculations
explained_var <- pca_result$sdev^2
prop_var <- explained_var / sum(explained_var)
cum_var <- cumsum(prop_var)

# Step 2: Create a data frame for plotting
```

```r
scree_data <- data.frame(
  PC = factor(paste0("PC", 1:length(prop_var)), levels = paste0("PC", 1:length(prop_var))),
  VarianceExplained = prop_var,
  CumulativeVariance = cum_var
)

# Step 3: Bar plot (first126 PCs for clarity)
ggplot(scree_data[1:126, ], aes(x = PC, y = VarianceExplained)) +
  geom_bar(stat = "identity", fill = "steelblue") +
  geom_hline(yintercept = 0.95, linetype = "dashed", color = "green") +
  theme_minimal() +
  labs(
    title = "Scree Plot (Bar) with Cumulative Variance Line",
    x = "Principal Components",
    y = "Proportion of Variance Explained"
  ) +
 scale_y_continuous(limits = c(0, 0.125)) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

```
## Warning: Removed 2 rows containing missing values or values outside the scale range
## (`geom_bar()`).
```

```
## Warning: Removed 1 row containing missing values or values outside the scale range
## (`geom_hline()`).
```



```r
# Option 2: just get number of total PCs
ncol(pca_result$x)
```

```
## [1] 198
```

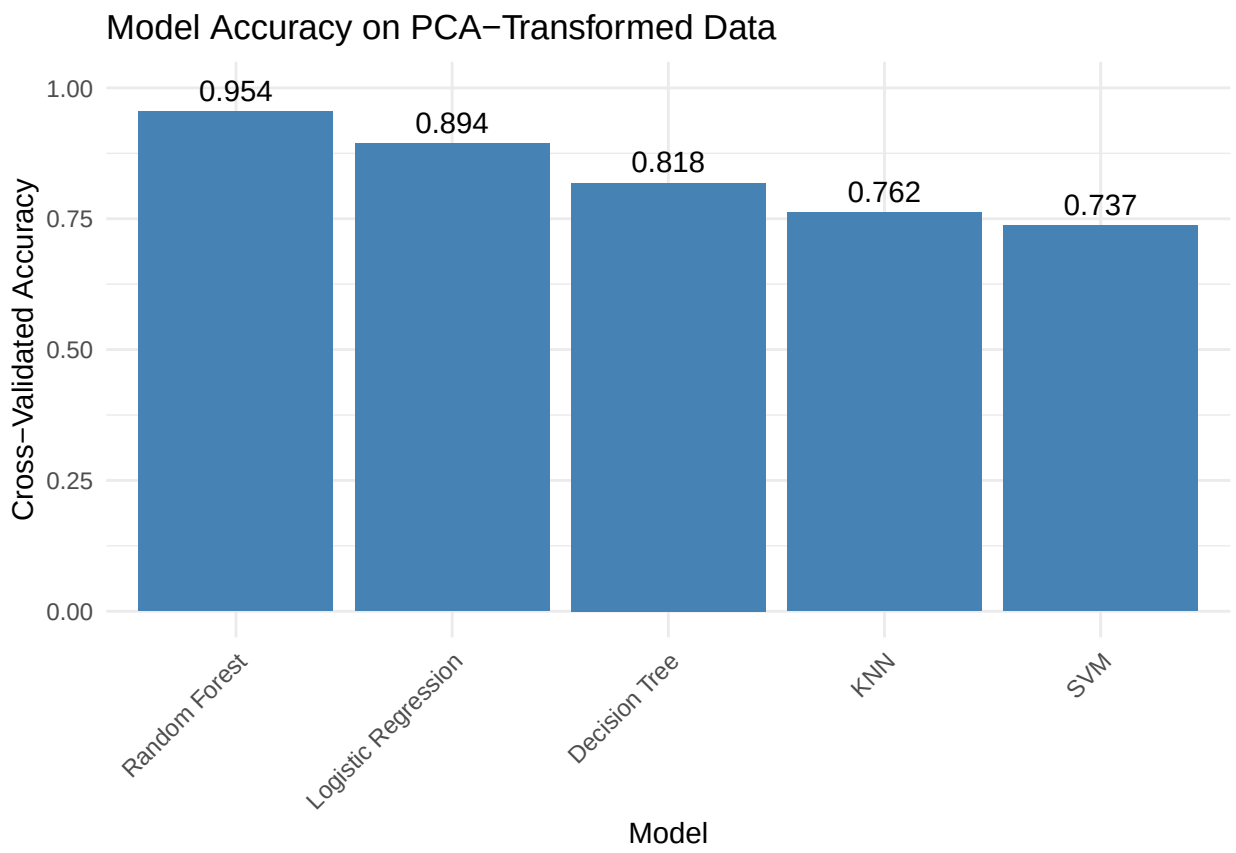
```r
# Project original data onto the top components
pca_95_data <- as.data.frame(pca_result$x[, 1:num_pc_95])

# Add back the labels
pca_95_data$target <- y_train_resampled  # Labels from upsampled data
```

```r
library(caret)
library(e1071)
library(randomForest)
library(rpart)
library(kknn)

# 1. Extract features and target from PCA data
X_pca <- pca_95_data[, -ncol(pca_95_data)]  # all PC columns
y_pca <- pca_95_data$target                 # target labels

# 2. Define 5-fold CV
ctrl <- trainControl(method = "cv", number = 5)

# 3. Train models using PCA data
set.seed(42)
model_results_pca <- list()

model_results_pca[["Logistic Regression"]] <- train(
  x = X_pca, y = y_pca,
  method = "multinom",
  trControl = ctrl,
  trace = FALSE
)

model_results_pca[["SVM"]] <- train(
  x = X_pca, y = y_pca,
  method = "svmLinear",
  trControl = ctrl
)

model_results_pca[["Decision Tree"]] <- train(
  x = X_pca, y = y_pca,
  method = "rpart",
  trControl = ctrl
)

model_results_pca[["Random Forest"]] <- train(
  x = X_pca, y = y_pca,
  method = "rf",
  trControl = ctrl
)
```

```r
model_results_pca[["KNN"]] <- train(
  x = X_pca, y = y_pca,
  method = "knn",
  trControl = ctrl,
  tuneLength = 5
)

# 4. Compare PCA-based model accuracy
model_accuracies_pca <- sapply(model_results_pca, function(m) max(m$results$Accuracy))
print(model_accuracies_pca)
```

```
## Logistic Regression                 SVM       Decision Tree       Random Forest
##           0.8937179           0.7371795           0.8182833           0.9594482
##                 KNN
##           0.7578317
```

```r
library(ggplot2)

# Create a named data frame from your accuracy results
model_accuracies_pca <- c(
  "Logistic Regression" = 0.8937179,
  "SVM" = 0.7371795 ,
  "Decision Tree" = 0.8182833,
  "Random Forest" = 0.9544482,
  "KNN" = 0.7616273
)

accuracy_df <- data.frame(
  Model = names(model_accuracies_pca),
  Accuracy = as.numeric(model_accuracies_pca)
)

# Plot
ggplot(accuracy_df, aes(x = reorder(Model, -Accuracy), y = Accuracy)) +
  geom_bar(stat = "identity", fill = "steelblue") +
  geom_text(aes(label = round(Accuracy, 3)), vjust = -0.5, size = 4) +
  ylim(0, 1) +
  theme_minimal() +
  labs(
    title = "Model Accuracy on PCA-Transformed Data",
    x = "Model",
    y = "Cross-Validated Accuracy"
  ) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```




```
library(yardstick)
```

```
## Warning: package 'yardstick' was built under R version 4.4.3
```

```
##
## Attaching package: 'yardstick'
```

```
## The following objects are masked from 'package:caret':
##
##     precision, recall, sensitivity, specificity
```

```
## The following object is masked from 'package:readr':
##
##     spec
```

```
library(dplyr)
library(pROC)

# Get predicted class labels and true labels
rf_preds <- predict(model_results_pca[["Random Forest"]], X_pca)
true_labels <- y_pca

# Convert to data frame for yardstick
eval_df <- data.frame(
```

```r
  truth = true_labels,
  prediction = rf_preds
)

# F1 (macro-averaged)
f1_macro <- eval_df %>%
  yardstick::f_meas(truth = truth, estimate = prediction, beta = 1)

# F2 (macro-averaged)
f2_macro <- eval_df %>%
  yardstick::f_meas(truth = truth, estimate = prediction, beta = 2)

print(f1_macro)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 f_meas  macro              1
```

```r
print(f2_macro)
```

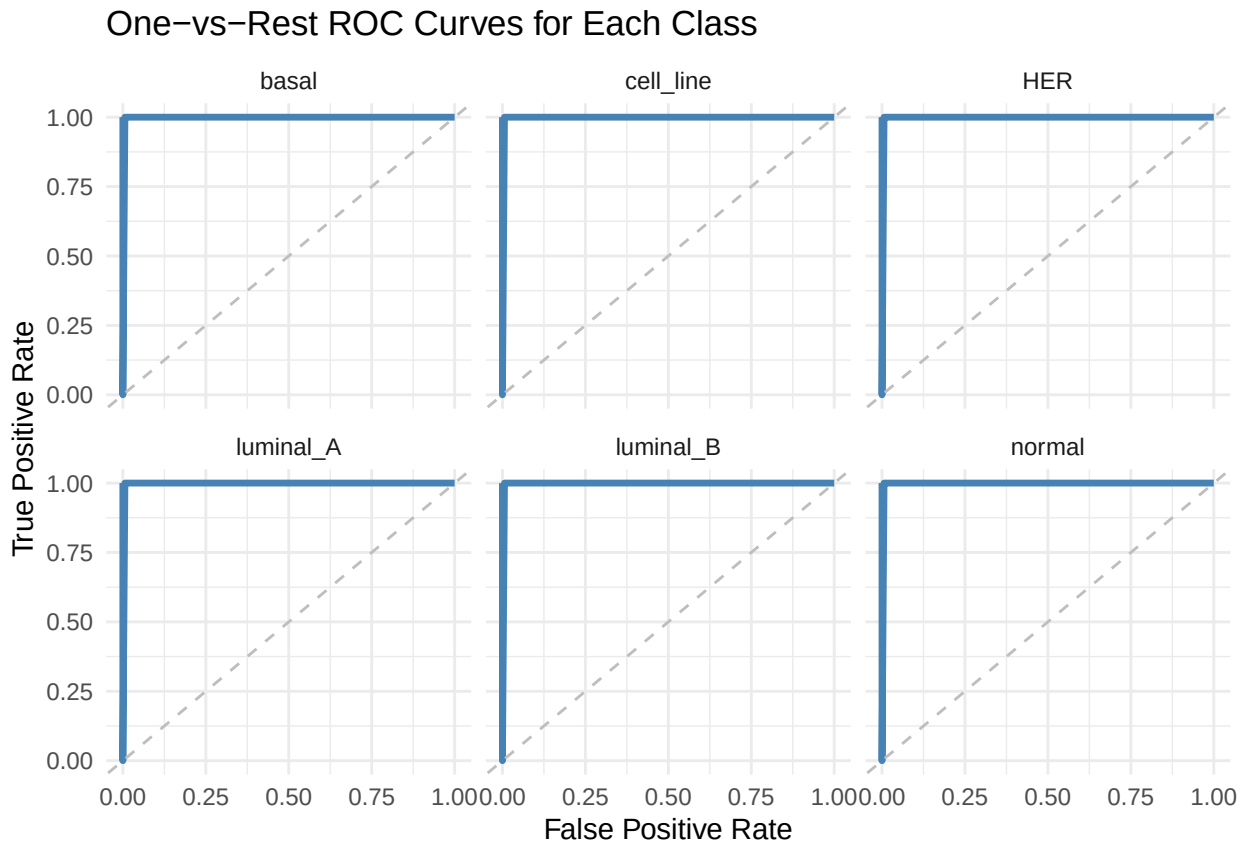
```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 f_meas  macro              1
```

```r
# Predict probabilities
rf_probs <- predict(model_results_pca[["Random Forest"]], X_pca, type = "prob")

# Use pROC's multiclass AUC
library(pROC)
roc_multiclass <- multiclass.roc(response = true_labels, predictor = as.matrix(rf_probs))
auc_value <- auc(roc_multiclass)
print(auc_value)
```

```
## Multi-class area under the curve: 1
```

```r
library(pROC)
library(ggplot2)

# 1. Get true labels and predicted probabilities
true_labels <- y_pca
rf_probs <- predict(model_results_pca[["Random Forest"]], X_pca, type = "prob")
class_levels <- colnames(rf_probs)

# 2. Compute one-vs-rest ROC for each class
roc_df <- do.call(rbind, lapply(class_levels, function(class) {
  binary_response <- as.numeric(true_labels == class)
  roc_obj <- roc(binary_response, rf_probs[[class]], quiet = TRUE)

  data.frame(
```

```
  fpr = 1 - roc_obj$specificities,
  tpr = roc_obj$sensitivities,
  class = class,
  auc = rep(auc(roc_obj), length(roc_obj$sensitivities))
)

}))

# 3. Plot with facets
ggplot(roc_df, aes(x = fpr, y = tpr)) +
  geom_line(linewidth = 1.2, color = "steelblue") +
  geom_abline(linetype = "dashed", color = "gray") +
  facet_wrap(~ class, ncol = 3) +
  theme_minimal() +
  labs(
    title = "One-vs-Rest ROC Curves for Each Class",
    x = "False Positive Rate",
    y = "True Positive Rate"
  )
```




```
# Get predictions on PCA data
rf_preds <- predict(model_results_pca[["Random Forest"]], X_pca)
library(caret)

# 1. Predict on PCA-transformed training data
```

```r
rf_preds <- predict(model_results_pca[["Random Forest"]], X_pca)

# 2. Generate confusion matrix
conf_matrix <- confusionMatrix(rf_preds, y_pca)

# 3. View class-wise performance
metrics_per_class <- conf_matrix$byClass
print(metrics_per_class)
```

```
##                  Sensitivity Specificity Pos Pred Value Neg Pred Value
## Class: basal               1           1              1              1
## Class: cell_line           1           1              1              1
## Class: HER                 1           1              1              1
## Class: luminal_A           1           1              1              1
## Class: luminal_B           1           1              1              1
## Class: normal              1           1              1              1
##                  Precision Recall F1 Prevalence Detection Rate
## Class: basal             1      1  1  0.1666667      0.1666667
## Class: cell_line         1      1  1  0.1666667      0.1666667
## Class: HER               1      1  1  0.1666667      0.1666667
## Class: luminal_A         1      1  1  0.1666667      0.1666667
## Class: luminal_B         1      1  1  0.1666667      0.1666667
## Class: normal            1      1  1  0.1666667      0.1666667
##                  Detection Prevalence Balanced Accuracy
## Class: basal                0.1666667                 1
## Class: cell_line            0.1666667                 1
## Class: HER                  0.1666667                 1
## Class: luminal_A            0.1666667                 1
## Class: luminal_B            0.1666667                 1
## Class: normal               0.1666667                 1
```

```r
library(randomForest)
library(caret)

# 1. Train RF on the PCA-scores of your resampled training set
set.seed(42)
rf_model <- randomForest(
  x = X_pca_train,
  y = y_train_resampled
)

# 2. Get predictions on both train and test
train_preds <- predict(rf_model, newdata = X_pca_train)
test_preds  <- predict(rf_model, newdata = df_pca_test[, grep("^PC", names(df_pca_test))])

# 3. Evaluate with confusion matrices
cat("\n--- Training Set Performance ---\n")
```

```
##
## --- Training Set Performance ---
```

```r
print(confusionMatrix(train_preds, y_train_resampled))
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  basal cell_line HER luminal_A luminal_B normal
##   basal        33         0   0         0         0      0
##   cell_line     0        33   0         0         0      0
##   HER           0         0  33         0         0      0
##   luminal_A     0         0   0        33         0      0
##   luminal_B     0         0   0         0        33      0
##   normal        0         0   0         0         0     33
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9815, 1)
##     No Information Rate : 0.1667
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: basal Class: cell_line Class: HER Class: luminal_A
## Sensitivity                1.0000           1.0000     1.0000           1.0000
## Specificity                1.0000           1.0000     1.0000           1.0000
## Pos Pred Value             1.0000           1.0000     1.0000           1.0000
## Neg Pred Value             1.0000           1.0000     1.0000           1.0000
## Prevalence                 0.1667           0.1667     0.1667           0.1667
## Detection Rate             0.1667           0.1667     0.1667           0.1667
## Detection Prevalence       0.1667           0.1667     0.1667           0.1667
## Balanced Accuracy          1.0000           1.0000     1.0000           1.0000
##                      Class: luminal_B Class: normal
## Sensitivity                    1.0000        1.0000
## Specificity                    1.0000        1.0000
## Pos Pred Value                 1.0000        1.0000
## Neg Pred Value                 1.0000        1.0000
## Prevalence                     0.1667        0.1667
## Detection Rate                 0.1667        0.1667
## Detection Prevalence           0.1667        0.1667
## Balanced Accuracy              1.0000        1.0000
```

```r
cat("\n--- Testing Set Performance ---\n")
```

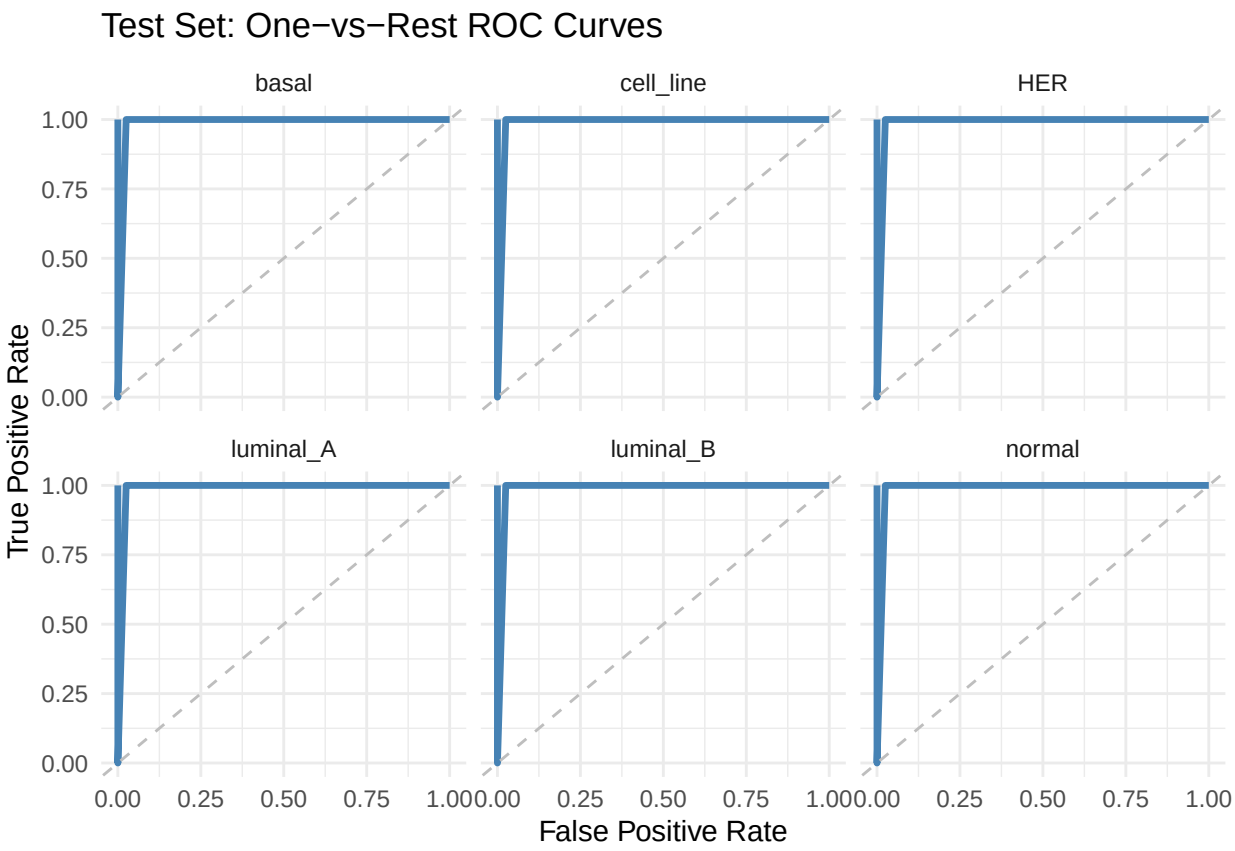
```
##
## --- Testing Set Performance ---
```

```r
print(confusionMatrix(test_preds, y_test_resampled))
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  basal cell_line HER luminal_A luminal_B normal
##   basal         7         0   0         0         0      0
##   cell_line     0         8   0         0         0      0
##   HER           1         0   8         0         0      0
##   luminal_A     0         0   0         8         1      0
##   luminal_B     0         0   0         0         7      0
##   normal        0         0   0         0         0      8
##
## Overall Statistics
##
##                Accuracy : 0.9583
##                  95% CI : (0.8575, 0.9949)
##     No Information Rate : 0.1667
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.95
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: basal Class: cell_line Class: HER Class: luminal_A
## Sensitivity                0.8750           1.0000     1.0000           1.0000
## Specificity                1.0000           1.0000     0.9750           0.9750
## Pos Pred Value             1.0000           1.0000     0.8889           0.8889
## Neg Pred Value             0.9756           1.0000     1.0000           1.0000
## Prevalence                 0.1667           0.1667     0.1667           0.1667
## Detection Rate             0.1458           0.1667     0.1667           0.1667
## Detection Prevalence       0.1458           0.1667     0.1875           0.1875
## Balanced Accuracy          0.9375           1.0000     0.9875           0.9875
##                      Class: luminal_B Class: normal
## Sensitivity                    0.8750        1.0000
## Specificity                    1.0000        1.0000
## Pos Pred Value                 1.0000        1.0000
## Neg Pred Value                 0.9756        1.0000
## Prevalence                     0.1667        0.1667
## Detection Rate                 0.1458        0.1667
## Detection Prevalence           0.1458        0.1667
## Balanced Accuracy              0.9375        1.0000
```

```r
library(pROC)
library(yardstick)
library(dplyr)
library(ggplot2)

# --- Predict class probabilities on the resampled PCA test set ---
rf_probs_test <- predict(
  rf_model,
  newdata = df_pca_test[, grep("^PC", names(df_pca_test))],
  type    = "prob"
)
```

```r
true_labels_test <- y_test_resampled

class_levels <- colnames(rf_probs_test)

# --- AUC ROC Computation ---
roc_df_test <- do.call(rbind, lapply(class_levels, function(class) {
  binary_response <- as.numeric(true_labels_test == class)
  roc_obj <- roc(binary_response, rf_probs_test[, class], quiet = TRUE)

  data.frame(
    fpr   = 1 - roc_obj$specificities,
    tpr   = roc_obj$sensitivities,
    class = class,
    auc   = rep(round(auc(roc_obj), 3), length(roc_obj$sensitivities))
  )
}))

# --- ROC PLOT ---
ggplot(roc_df_test, aes(x = fpr, y = tpr)) +
  geom_line(linewidth = 1.2, color = "steelblue") +
  geom_abline(linetype = "dashed", color = "gray") +
  facet_wrap(~ class, ncol = 3) +
  theme_minimal() +
  labs(
    title = "Test Set: One-vs-Rest ROC Curves",
    x     = "False Positive Rate",
    y     = "True Positive Rate"
  )
```


Test Set: One−vs−Rest ROC Curves
basal
cell_line
HER
luminal_A
luminal_B
normal
True Positive Rate
False Positive Rate
0.00
0.25
0.50
0.75
1.00