BIOSTAT C161 HW2

:)

Hanxi Chen

2025-10-23

```r
library(tidyverse)
```

```
Warning: package 'ggplot2' was built under R version 4.4.3
```

```r
library(dplyr)
library(caret)
```

Q2

This problem uses the Weekly.csv dataset (uploaded on Bruinlearn) containing 1089 weekly stock returns for 21 years ### (a) Use the full dataset to fit a logistic regression of today's stock movement (up or down) on the five lags of returns and the trading volume.

Sol:

```r
# setwd('D:/R/BIOSTAT M236/BIOSTAT_M236_longitudinal/hw2')

Weekly <- read.csv("./Weekly.csv")
head(Weekly)
```

```
  Year   Lag1   Lag2   Lag3   Lag4   Lag5    Volume  Today Direction
1 1990  0.816  1.572 -3.936 -0.229 -3.484 0.1549760 -0.270      Down
2 1990 -0.270  0.816  1.572 -3.936 -0.229 0.1485740 -2.576      Down
3 1990 -2.576 -0.270  0.816  1.572 -3.936 0.1598375  3.514        Up
4 1990  3.514 -2.576 -0.270  0.816  1.572 0.1616300  0.712        Up
5 1990  0.712  3.514 -2.576 -0.270  0.816 0.1537280  1.178        Up
6 1990  1.178  0.712  3.514 -2.576 -0.270 0.1544440 -1.372      Down
```

```r
Weekly$Direction <- as.factor(Weekly$Direction)

model_full <- glm(Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume,
                  data = Weekly,
                  family = binomial)
```

```r
summary(model_full)
```

```
Call:
glm(formula = Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 +
    Volume, family = binomial, data = Weekly)

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  0.26686    0.08593   3.106   0.0019 **
Lag1        -0.04127    0.02641  -1.563   0.1181
Lag2         0.05844    0.02686   2.175   0.0296 *
Lag3        -0.01606    0.02666  -0.602   0.5469
Lag4        -0.02779    0.02646  -1.050   0.2937
Lag5        -0.01447    0.02638  -0.549   0.5833
Volume      -0.02274    0.03690  -0.616   0.5377
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1496.2  on 1088  degrees of freedom
Residual deviance: 1486.4  on 1082  degrees of freedom
AIC: 1500.4

Number of Fisher Scoring iterations: 4
```

(b)

Calculate the confusion matrix, accuracy, precision, recall, and F1 score for the in-sample predictions. Does the model uniformly beat random guessing in terms of these performance metrics?

Sol:

```r
prob_pred <- predict(model_full, type = "response")

pred_class <- ifelse(prob_pred > 0.5, "Up", "Down")
pred_class <- factor(pred_class, levels = levels(Weekly$Direction))

# Confusion matrix
conf_mat <- confusionMatrix(pred_class, Weekly$Direction)
conf_mat
```

```
Confusion Matrix and Statistics
```

```
          Reference
Prediction Down  Up
      Down   54  48
      Up    430 557

               Accuracy : 0.5611
                 95% CI : (0.531, 0.5908)
    No Information Rate : 0.5556
    P-Value [Acc > NIR] : 0.369

                  Kappa : 0.035

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.11157
            Specificity : 0.92066
         Pos Pred Value : 0.52941
         Neg Pred Value : 0.56434
             Prevalence : 0.44444
         Detection Rate : 0.04959
   Detection Prevalence : 0.09366
      Balanced Accuracy : 0.51612

       'Positive' Class : Down
```

```r
table_pred <- table(Predicted = pred_class, Actual = Weekly$Direction)
table_pred
```

```
         Actual
Predicted Down  Up
     Down   54  48
     Up    430 557
```

```r
# Extract metrics manually
TP <- table_pred["Up", "Up"]
TN <- table_pred["Down", "Down"]
FP <- table_pred["Up", "Down"]
FN <- table_pred["Down", "Up"]

accuracy  <- (TP + TN) / sum(table_pred)
precision <- TP / (TP + FP)
recall    <- TP / (TP + FN)
F1        <- 2 * precision * recall / (precision + recall)

# Print metrics
cat("Accuracy:", round(accuracy, 3), "\n")
```

```
Accuracy: 0.561
```

```r
cat("Precision:", round(precision, 3), "\n")
```

```
Precision: 0.564
```

```r
cat("Recall:", round(recall, 3), "\n")
```

```
Recall: 0.921
```

```r
cat("F1 Score:", round(F1, 3), "\n")
```

```
F1 Score: 0.7
```

For random guessing with an equal number of "Up" and "Down" observations, the expected accuracy, precision, recall, and F1 score are all 0.5. In comparison, the fitted logistic regression model yields an accuracy of 0.561, precision of 0.564, recall of 0.921, and F1 score of 0.700. Although the model slightly outperforms random guessing across all metrics, the improvement is modest and largely driven by its strong tendency to predict "Up," which inflates recall. Therefore, while the model performs better than random guessing numerically, it does not provide substantial predictive power in practice.

(c)

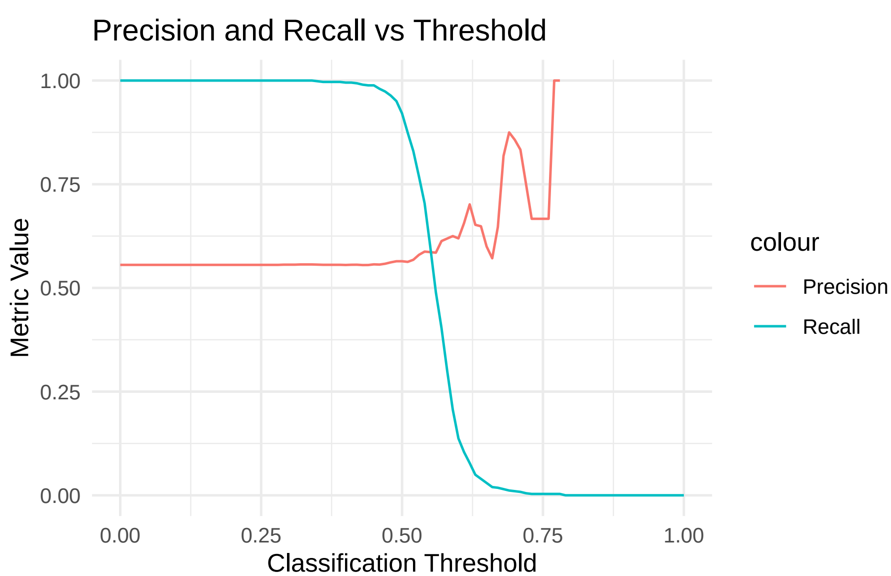
On the same graph, plot precision and recall against the threshold (varying over [0, 1]) used to generate predicted labels from predicted probabilities. Explain the pattern you see

Sol:

```r
thresholds <- seq(0, 1, by = 0.01)

precision_vals <- numeric(length(thresholds))
recall_vals <- numeric(length(thresholds))

for (i in seq_along(thresholds)) {
  t <- thresholds[i]
  pred_t <- ifelse(prob_pred > t, "Up", "Down")
  pred_t <- factor(pred_t, levels = levels(Weekly$Direction))
  cm <- table(Predicted = pred_t, Actual = Weekly$Direction)

  # Extract TP, FP, FN
  TP <- cm["Up", "Up"]
  FP <- cm["Up", "Down"]
  FN <- cm["Down", "Up"]

  precision_vals[i] <- TP / (TP + FP)
  recall_vals[i] <- TP / (TP + FN)
}
```

```r
pr_df <- data.frame(threshold = thresholds,
                    Precision = precision_vals,
                    Recall = recall_vals)

ggplot(pr_df, aes(x = threshold)) +
  geom_line(aes(y = Precision, color = "Precision")) +
  geom_line(aes(y = Recall, color = "Recall")) +
  labs(title = "Precision and Recall vs Threshold",
       x = "Classification Threshold",
       y = "Metric Value") +
  theme_minimal()
```

```
Warning: Removed 22 rows containing missing values or values outside the scale range
(`geom_line()`).
```



In this plot, precision and recall exhibit opposite trends as the classification threshold changes. When the threshold is low, almost all observations are predicted as "Up," so recall remains close to 1 while precision stays around 0.5 because many "Down" cases are incorrectly labeled as "Up." As the threshold increases, fewer observations are classified as "Up," causing recall to drop sharply while precision gradually rises, reflecting fewer false positives. The fluctuations in precision at higher thresholds occur because very few positive predictions are made, so small changes in predictions can cause large variations in precision. Overall, the pattern illustrates the fundamental trade-off between precision and recall in binary classification.

(d)

Now fit the logistic regression using only data up to (and including) the year 2008, with Lag2 as the only predictor.

Sol:

```r
train_data <- subset(Weekly, Year <= 2008)

model_lag2 <- glm(Direction ~ Lag2, data = train_data, family = binomial)

summary(model_lag2)
```

```

Call:
glm(formula = Direction ~ Lag2, family = binomial, data = train_data)

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  0.20326    0.06428   3.162  0.00157 **
Lag2         0.05810    0.02870   2.024  0.04298 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1354.7  on 984  degrees of freedom
Residual deviance: 1350.5  on 983  degrees of freedom
AIC: 1354.5

Number of Fisher Scoring iterations: 4
```

(e)

Repeat (b) using the remaining observations as a test sample.

Sol:

```r
test_data <- subset(Weekly, Year > 2008)

prob_test <- predict(model_lag2, newdata = test_data, type = "response")

pred_test <- ifelse(prob_test > 0.5, "Up", "Down")

table(Predicted = pred_test, Actual = test_data$Direction)
```

```
         Actual
Predicted Down Up
     Down    9  5
```

```
    Up     34 56
```

```r
accuracy <- mean(pred_test == test_data$Direction)

precision <- sum(pred_test == "Up" & test_data$Direction == "Up") /
             sum(pred_test == "Up")

recall <- sum(pred_test == "Up" & test_data$Direction == "Up") /
          sum(test_data$Direction == "Up")

f1 <- 2 * precision * recall / (precision + recall)

cat("Accuracy:", round(accuracy, 3), "\n")
```

```
Accuracy: 0.625
```

```r
cat("Precision:", round(precision, 3), "\n")
```

```
Precision: 0.622
```

```r
cat("Recall:", round(recall, 3), "\n")
```

```
Recall: 0.918
```

```r
cat("F1 Score:", round(f1, 3), "\n")
```

```
F1 Score: 0.742
```

The model trained using data up to 2008 and tested on later years achieves an accuracy of 0.625, meaning it correctly predicts about 62.5% of weekly market directions. Its precision of 0.622 indicates that when the model predicts "Up," it is correct about 62% of the time, while the high recall of 0.918 shows that it successfully identifies around 92% of all actual "Up" weeks. The F1 score of 0.742, which balances precision and recall, suggests overall good predictive consistency. Compared to random guessing (which would yield about 0.5 for accuracy, precision, recall, and F1), this model performs notably better, particularly in recall. However, the model tends to overpredict "Up" movements, which explains its high recall but moderate precision.

(f)

Which of the two fitted models would you use for real-time stock return prediction?

Sol: For real-time stock return prediction, the model trained only on data up to 2008 (the Lag2 model) would be preferred over the model trained on the full dataset. The reason is that this model simulates a true forecasting scenario—using only past information to predict future outcomes—making its performance on post-2008 data a more realistic measure of predictive ability. Although its accuracy (0.625) is modest, it outperforms random guessing and demonstrates generalization beyond the training sample. In contrast, the full-sample model evaluates in-sample performance and thus may suffer from overfitting, offering an overly optimistic view of its predictive power in real-world applications.