AMRC: Exercise 5 - 11912007

Analysis

From the analysis of the loan dataset using least squares regression, several key findings emerge. The data required preprocessing before model fitting, including encoding categorical variables (Status, EmpLen, and Home) into numeric format and scaling continuous variables like Amount and Income.

The model summary reveals a residual standard error of 0.3497 with 589 degrees of freedom, and an F-statistic of 4.799 with a highly significant p-value of 0.000001193, indicating that the model has some predictive power / is competitive. However, three coefficients were not defined due to singularities, suggesting potential multicollinearity issues in the predictors.

When examining the model's performance, we found that using different cutoff values yielded varying results. The default cutoff achieved 87% accuracy but with extremely imbalanced sensitivity (1.00) and specificity (0.00), indicating poor classification balance. A balanced cutoff of approximately 0.83 was identified as optimal, providing more balanced performance metrics - slightly better than the Youden's J statistic cutoff.

The ROC analysis showed an Area Under Curve (AUC) of 0.6963 for the training set and 0.645 for the test set. These values, while better than random chance (0.5), indicate only moderate classifier performance. The confusion matrices revealed that with the balanced cutoff, the model correctly identified 58 true negatives and 317 true positives in the training set, but also produced 34 false positives and 191 false negatives.

The visualization of predictions against true class labels demonstrated clear overlap between the classes, making perfect separation impossible with this linear model. This is further supported by the TPR (True Positive Rate) and TNR (True Negative Rate) curves, which show the trade-off between sensitivity and specificity across different cutoff values.

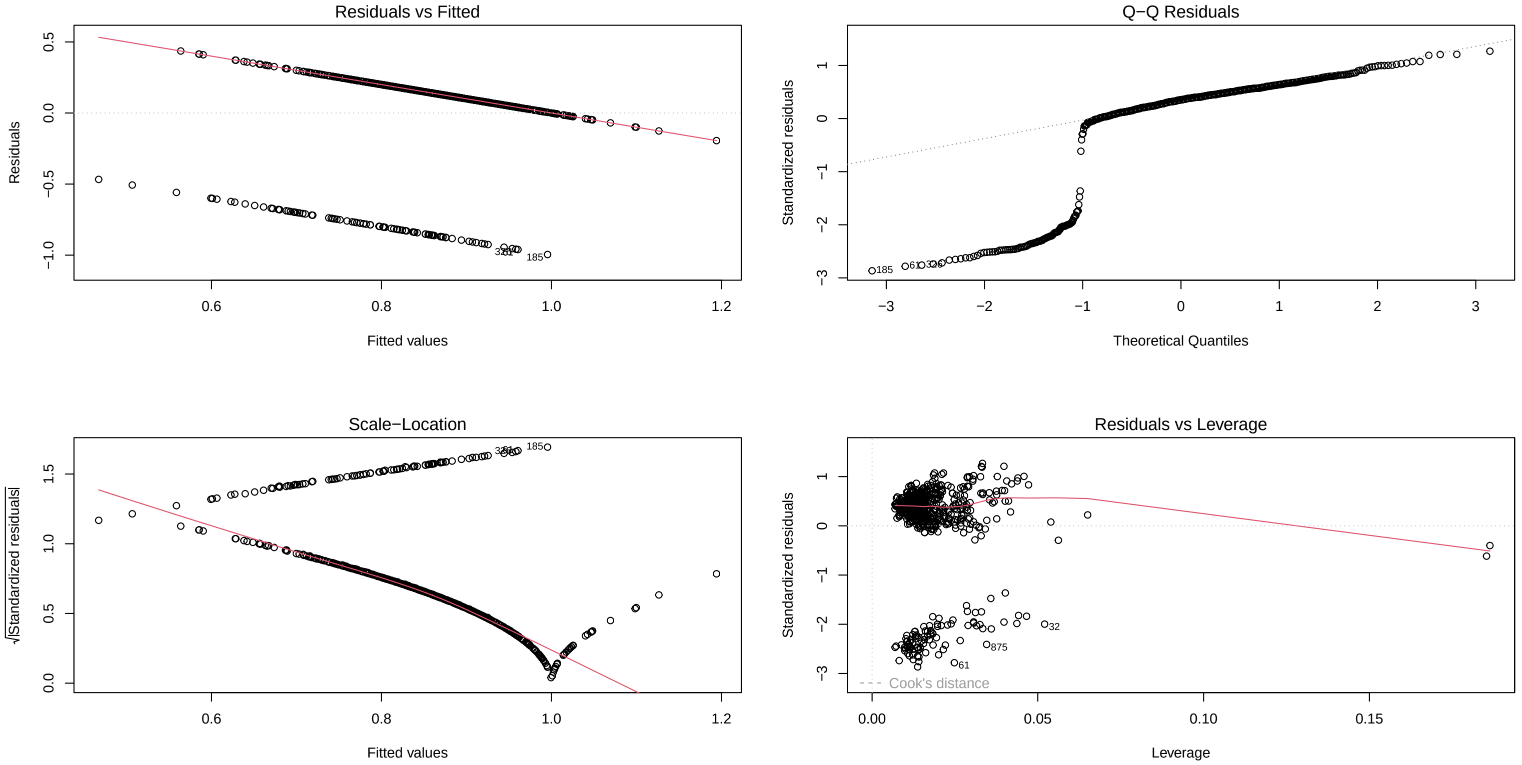
The final test set results confirm the moderate performance of the classifier, with similar patterns observed in both training and test sets. The imbalanced nature of the dataset (with significantly more positive cases than negative ones) presents a challenge for classification, as noted in the assignment remarks. While the least squares regression approach provides some predictive capability, there is room for improvement through more sophisticated classification methods and better handling of class imbalance.

Code

```r
#
# 1) preprocess, fit
#

# encoding categorical variables
Loan$Status <- as.numeric(Loan$Status == "FP")

Loan$EmpLen_A <- as.numeric(Loan$EmpLen == "A")
Loan$EmpLen_B <- as.numeric(Loan$EmpLen == "B")
Loan$EmpLen_C <- as.numeric(Loan$EmpLen == "C")
Loan$EmpLen_D <- as.numeric(Loan$EmpLen == "D")
Loan$EmpLen_U <- as.numeric(Loan$EmpLen == "U")
Loan$EmpLen <- NULL

Loan$Home_MORTGAGE <- as.numeric(Loan$Home == "MORTGAGE")
Loan$Home_OWN <- as.numeric(Loan$Home == "OWN")
Loan$Home_RENT <- as.numeric(Loan$Home == "RENT")
Loan$Home <- NULL

invisible(assert_that(all(sapply(Loan, is.numeric))))

# no missing values
invisible(assert_that(all(sapply(Loan, function(x) sum(is.na(x)) == 0))))

# Term col has only 1 unique value
invisible(assert_that(length(unique(Loan$Term)) == 1))
Loan$Term <- NULL

# scale columns that aren't encoded
Loan$Amount <- scale(Loan$Amount)
Loan$IntRate <- scale(Loan$IntRate)
Loan$Income <- scale(Loan$Income)
Loan$Score <- scale(Loan$Score)

# split
train_idx <- sample(nrow(Loan), size = round(2/3 * nrow(Loan)))
train_data <- Loan[train_idx, ]
test_data <- Loan[-train_idx, ]

# fit
model <- lm(Status ~ ., data = train_data)

#
# 2) analyze summary
#

summary(model)
```

```
##
## Call:
## lm(formula = Status ~ ., data = train_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.99540  0.02886  0.12231  0.19143  0.43622
##
## Coefficients: (3 not defined because of singularities)
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -1.818848   1.676842  -1.085  0.27850
## Amount        -0.027352   0.016485  -1.659  0.09761 .
## IntRate       -0.216395   0.093256  -2.320  0.02066 *
## ILR           74.554545  49.949941   1.493  0.13608
## Income         0.021565   0.015914   1.355  0.17591
## Score                NA         NA      NA       NA
## EmpLen_A       0.176350   0.065104   2.709  0.00695 **
## EmpLen_B       0.162601   0.065535   2.481  0.01337 *
## EmpLen_C       0.135204   0.067936   1.990  0.04703 *
## EmpLen_D       0.172160   0.062858   2.739  0.00635 **
## EmpLen_U             NA         NA      NA       NA
## Home_MORTGAGE  0.006562   0.031781   0.206  0.83648
## Home_OWN       0.065080   0.050335   1.293  0.19654
## Home_RENT            NA         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3497 on 589 degrees of freedom
## Multiple R-squared:  0.07533,    Adjusted R-squared:  0.05963
## F-statistic: 4.799 on 10 and 589 DF,  p-value: 0.000001193
```

```r
#
# 3) analyze plot
#

par(mfrow=c(2,2))
plot(model)
```




```r
#
# 4, 7) predict train set, find cutoff
#

preds <- predict(model, train_data)

pred <- prediction(preds, train_data$Status) # ROCR prediction object

# find cutoff that maximizes accuracy
perf_tpr_fpr <- performance(pred, "tpr", "fpr")
perf_acc <- performance(pred, "acc")
acc_cutoff <- perf_acc@x.values[[1]][which.max(perf_acc@y.values[[1]])]

# find cutoff that minimizes difference between sensitivity and specificity
perf_sens <- performance(pred, "sens")
perf_spec <- performance(pred, "spec")
sens_spec_diff <- abs(unlist(perf_sens@y.values) - unlist(perf_spec@y.values))
balanced_cutoff <- perf_sens@x.values[[1]][which.min(sens_spec_diff)]

# youden's J statistic (sensitivity + specificity - 1)
youden <- unlist(perf_sens@y.values) + unlist(perf_spec@y.values) - 1
youden_cutoff <- perf_sens@x.values[[1]][which.max(youden)]

evaluate_cutoff <- function(cutoff, predictions, actual) {
    predicted_classes <- ifelse(predictions > cutoff, 1, 0)
    tp <- sum(predicted_classes == 1 & actual == 1)
    tn <- sum(predicted_classes == 0 & actual == 0)
    fp <- sum(predicted_classes == 1 & actual == 0)
    fn <- sum(predicted_classes == 0 & actual == 1)

    accuracy <- (tp + tn) / (tp + tn + fp + fn)
    sensitivity <- tp / (tp + fn)
    specificity <- tn / (tn + fp)
    return(c(accuracy = accuracy, sensitivity = sensitivity, specificity = specificity))
}

test_preds <- predict(model, test_data)
cutoffs <- c(0.5, acc_cutoff, balanced_cutoff, youden_cutoff)
names(cutoffs) <- c("Default", "Max Accuracy", "Balanced", "Youden")

results <- sapply(cutoffs, evaluate_cutoff, predictions = test_preds, actual = test_data$Status)
colnames(results) <- names(cutoffs)

# results as kable
results_df <- as.data.frame(round(results, 4))
results_df$Metric <- rownames(results_df)
results_df <- results_df[, c(5, 1:4)]
knitr::kable(results_df, col.names = c("Metric", "Default", "Max Accuracy", "Balanced", "Youden"), align = c('l', rep('r', 4)), digits = 4, caption = "Performance Metrics for Differen
```

Table 1: Performance Metrics for Different Cutoff Values

	Metric	Default	Max Accuracy	Balanced	Youden
accuracy	accuracy	0.87	0.8467	0.6067	0.4300
sensitivity	sensitivity	1.00	0.9693	0.6054	0.3755
specificity	specificity	0.00	0.0256	0.6154	0.7949

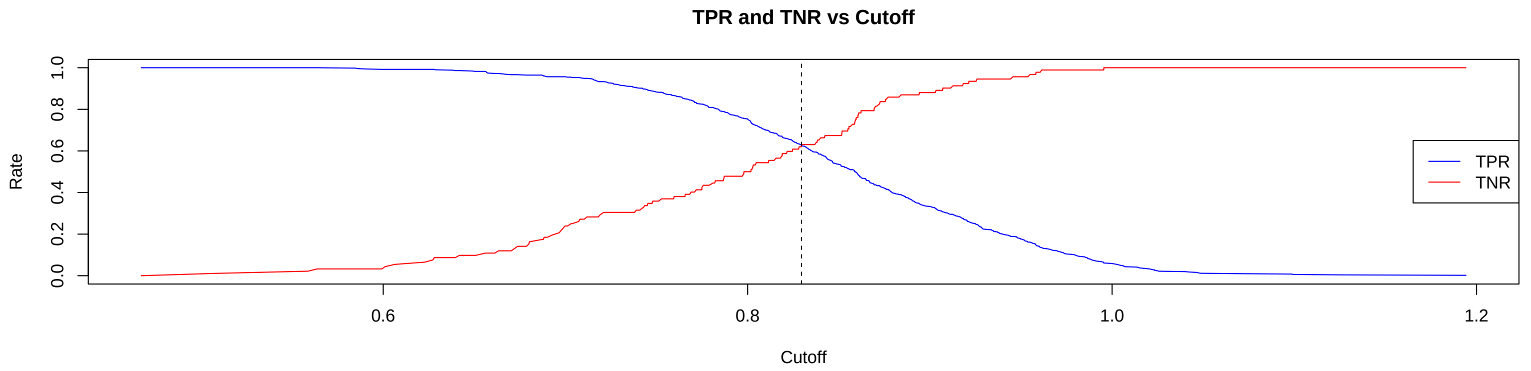
```r
# make sure the same result for both packages
train_measure <- measureit(score = preds, class = train_data$Status, measure = c("TPR", "TNR"))
test_measure <- measureit(score = test_preds, class = test_data$Status, measure = c("TPR", "TNR"))
measure_df <- data.frame(cutoff = train_measure$Cutoff, TPR = train_measure$TPR, TNR = train_measure$TNR, diff = abs(train_measure$TPR - train_measure$TNR))
measureit_balanced_cutoff <- measure_df$cutoff[which.min(measure_df$diff)]
cat("balanced cutoff from ROCR:", balanced_cutoff, "\n")
```

```
## balanced cutoff from ROCR: 0.8295281
```

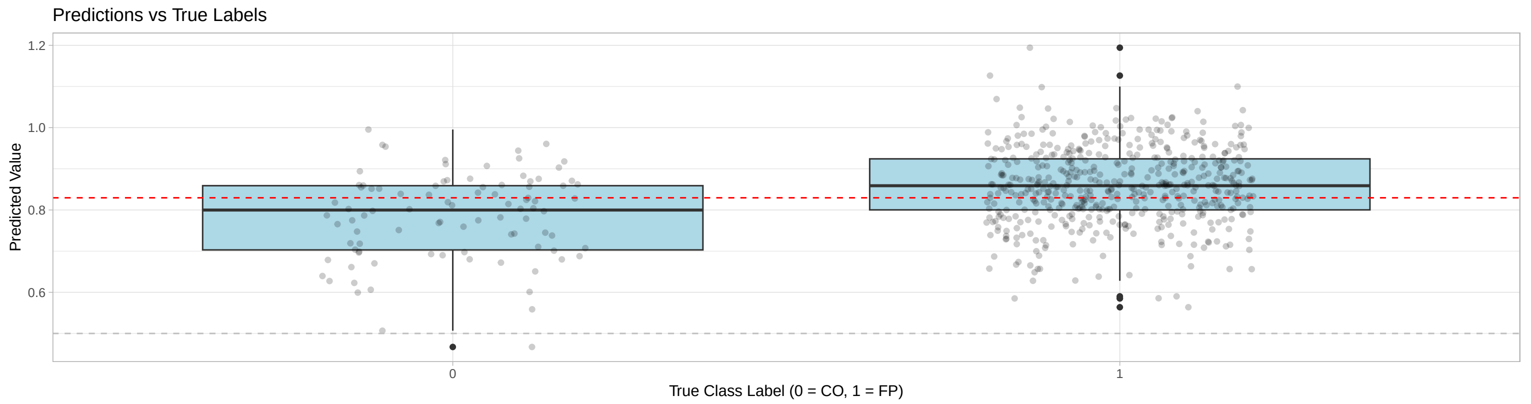
```r
cat("balanced cutoff from ROCit:", measureit_balanced_cutoff, "\n")
```

```
## balanced cutoff from ROCit: 0.8295281
```

```r
# plot the TPR and TNR curves
plot(train_measure$Cutoff, train_measure$TPR, type = "l", col = "blue", xlab = "Cutoff", ylab = "Rate", main = "TPR and TNR vs Cutoff")
lines(train_measure$Cutoff, train_measure$TNR, col = "red")
abline(v = measureit_balanced_cutoff, lty = 2)
legend("right", legend = c("TPR", "TNR"), col = c("blue", "red"), lty = 1)
```




```
ggplot(data.frame(true_label = train_data$Status, prediction = preds), aes(x = factor(true_label), y = prediction)) +
    geom_boxplot(fill = "lightblue") +
    geom_jitter(alpha = 0.2, width = 0.2) +
    labs(x = "True Class Label (0 = CO, 1 = FP)", y = "Predicted Value", title = "Predictions vs True Labels") +
    geom_hline(yintercept = balanced_cutoff, color = "red", linetype = "dashed") +
    geom_hline(yintercept = 0.5, color = "gray", linetype = "dashed") +
    theme_light()
```




```
#
# 5, 8) confusion matrix
#

train_predictions <- ifelse(preds > balanced_cutoff, 1, 0)
train_conf_matrix <- table(Actual = train_data$Status, Predicted = train_predictions)
print(train_conf_matrix)
```

```
##       Predicted
## Actual   0   1
##      0  58  34
##      1 191 317
```

```
test_predictions <- ifelse(test_preds > balanced_cutoff, 1, 0)
test_conf_matrix <- table(Actual = test_data$Status, Predicted = test_predictions)
print(train_conf_matrix)
```

```
##       Predicted
## Actual   0   1
##      0  58  34
##      1 191 317
```

```
#
# 6) rocit
#

train_roc <- rocit(score = preds, class = train_data$Status)
summary(train_roc)
```

```
##
##  Method used: empirical
##  Number of positive(s): 508
##  Number of negative(s): 92
##  Area under curve: 0.6963
```

```
test_preds <- predict(model, test_data)
test_roc <- rocit(score = test_preds, class = test_data$Status)
summary(test_roc)
```

```
##
##  Method used: empirical
##  Number of positive(s): 261
##  Number of negative(s): 39
##  Area under curve: 0.645
```

```
plot(test_roc)
```