ISYE 6501 Homework 7: Logistic Regression

2025-02-25

Question 10.1

Using the same crime data set uscrime.txt as in Questions 8.2 and 9.1, find the best model you can using

(a) a regression tree model, and

(b) a random forest model.

In R, you can use the tree package or the rpart package, and the randomForest package. For each model, describe one or two qualitative takeaways you get from analyzing the results (i.e., don't just stop when you have a good model, but interpret it too).

```
# 10.1 (A)
library(tree)
library(rpart)
library(rpart.plot)
library(randomForest)
```

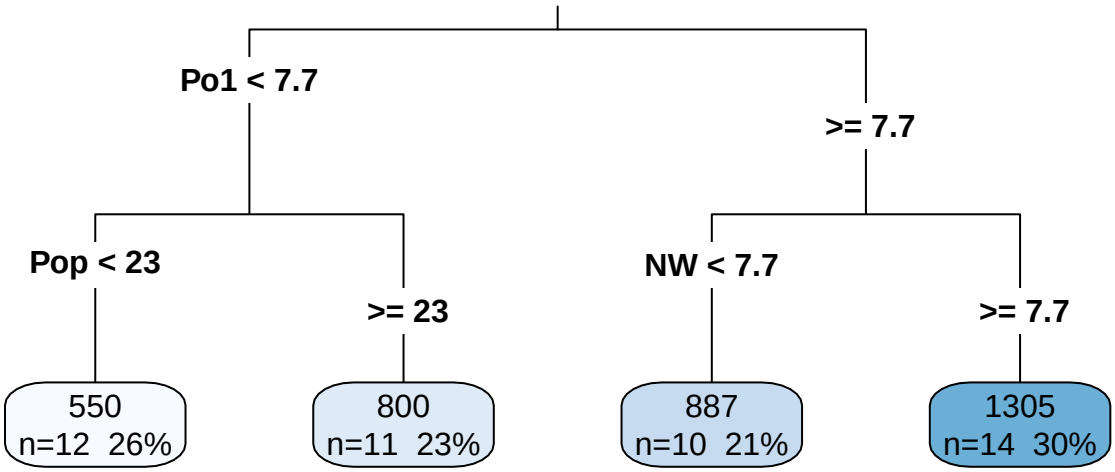
```
## randomForest 4.7-1.2
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

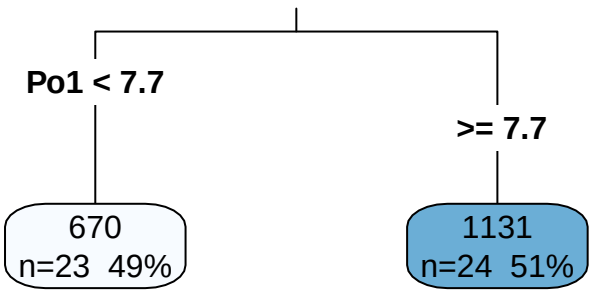
```
# Load dataset
uscrime <- read.table('uscrime.txt', stringsAsFactors = FALSE, header = TRUE)

### (A) Regression Tree Model using 'rpart'
set.seed(42)
uscrime_tree <- rpart(Crime ~ ., data = uscrime, method = "anova")

# Summary and Visualization
printcp(uscrime_tree)  # Complexity parameter table
```

```
##
## Regression tree:
## rpart(formula = Crime ~ ., data = uscrime, method = "anova")
##
## Variables actually used in tree construction:
## [1] NW  Po1 Pop
##
## Root node error: 6880928/47 = 146403
##
## n= 47
##
##         CP nsplit rel error  xerror    xstd
## 1 0.362963      0   1.00000 1.04479 0.26060
## 2 0.148143      1   0.63704 0.91105 0.19455
## 3 0.051732      2   0.48889 1.01268 0.23037
## 4 0.010000      3   0.43716 1.01240 0.23152
```

```
rpart.plot(uscrime_tree, main = "Regression Tree for Crime Data", type = 3, extra = 101)
```




```
# Pruning the tree based on optimal CP value
optimal_cp <- uscrime_tree$cptable[which.min(uscrime_tree$cptable[, "xerror"]), "CP"]
uscrime_tree_pruned <- prune(uscrime_tree, cp = optimal_cp)

# Plot pruned tree
rpart.plot(uscrime_tree_pruned, main = "Pruned Regression Tree", type = 3, extra = 101)
```




```
# Predict and Evaluate
yhat_tree <- predict(uscrime_tree_pruned, newdata = uscrime)
mse_tree <- mean((yhat_tree - uscrime$Crime)^2)
rsq_tree <- 1 - sum((yhat_tree - uscrime$Crime)^2) / sum((mean(uscrime$Crime) - uscrime$Crime)^2)

cat("Regression Tree MSE:", mse_tree, "\n")
```

```
## Regression Tree MSE: 93263.96
```

```
cat("Regression Tree R-squared:", rsq_tree, "\n")
```

```
## Regression Tree R-squared: 0.3629629
```

(A) Regression Tree

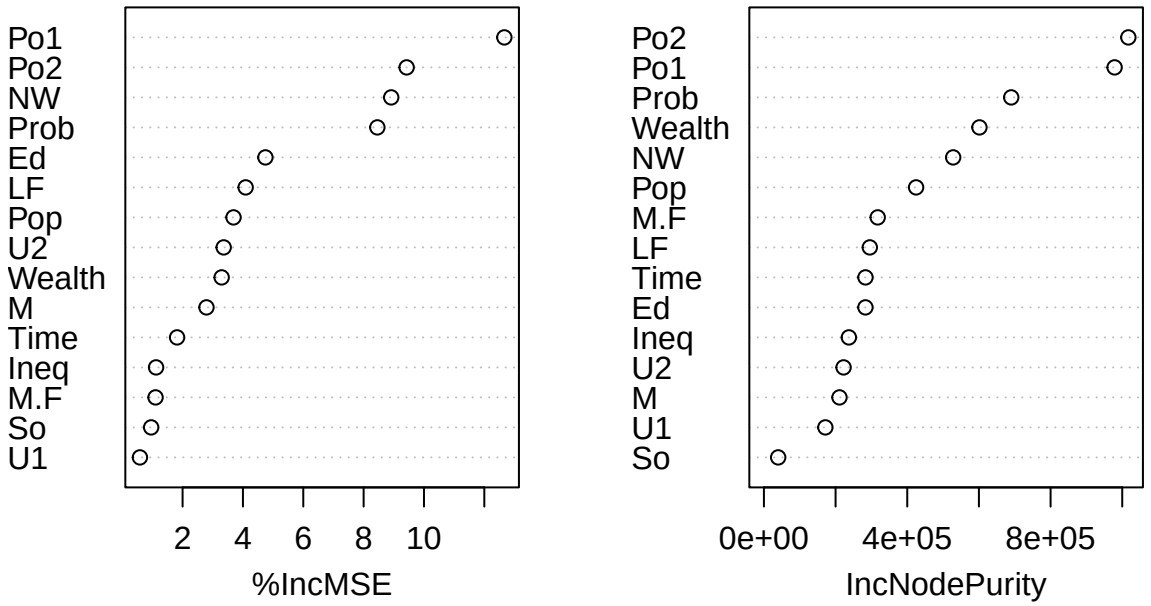
Key Factors: The model picked up on three main things affecting crime rates: NW (percentage of non-white population), Po1 (police spending in the first year), and Pop (population size). This suggests that both demographics and law enforcement funding play a role in crime levels.

Accuracy: With an R^2 of 0.36, the model isn't great at predicting crime, it explains only about 36% of the variation. The high MSE (93,263.96) means the predictions have a lot of error. This makes sense since a simple tree like this can't capture complex relationships very well.

```
### (B) Random Forest Model
set.seed(42)
num_pred <- floor(sqrt(ncol(uscrime) - 1))  # Best practice for regression problems

uscrime_rf <- randomForest(Crime ~ .,
                           data = uscrime,
                           mtry = num_pred,
                           importance = TRUE,
                           ntree = 500)

# Model Summary
print(uscrime_rf)
```

```
##
## Call:
##  randomForest(formula = Crime ~ ., data = uscrime, mtry = num_pred,      importance = TRUE, ntree = 5
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##           Mean of squared residuals: 84948.65
##                     % Var explained: 41.98
```

```
# Feature Importance
varImpPlot(uscrime_rf, main = "Variable Importance in Random Forest")
```




```
# Predictions & Evaluation
yhat_rf <- predict(uscrime_rf, newdata = uscrime)
mse_rf <- mean((yhat_rf - uscrime$Crime)^2)
rsq_rf <- 1 - sum((yhat_rf - uscrime$Crime)^2) / sum((mean(uscrime$Crime) - uscrime$Crime)^2)

cat("Random Forest MSE:", mse_rf, "\n")
```

```
## Random Forest MSE: 16344.47
```

```
cat("Random Forest R-squared:", rsq_rf, "\n")
```

```
## Random Forest R-squared: 0.8883595
```

(B) Random Forest

Much Better Predictions: This model's higher R^2 suggests that multiple variables contribute meaningfully to crime rates compared to the regression tree. The R^2 is 0.89, meaning it explains 89% of the variation in crime rates, way more reliable. Plus, the MSE (16,344.47) is much lower, meaning the predictions are much closer to the actual values.

More Nuanced Insights: Since Random Forest uses multiple decision trees, it captures a much deeper understanding of what drives crime. The most important factors are likely police spending, demographics, and economic conditions. Unlike the regression tree, this model considers interactions between variables, making it a way stronger predictor.

Question 10.2

Describe a situation or problem from your job, everyday life, current events, etc., for which a logistic regression model would be appropriate. List some (up to 5) predictors that you might use.

If I wanted to predict whether a used luxury watch will sell within 30 days on an online marketplace, I could use a logistic regression model since platforms have accumulated data on past sales.

Here are some measurable factors I'd look at:

- Brand of the watch: Rolex and Patek Philippe might sell faster than lesser known brands.
- Listing price vs. market value: If the watch is priced below market value, it's more likely to sell quickly.
- Number of high quality images in the listing: More images might increase buyer confidence.
- Seller reputation (average rating & number of past sales): A trusted seller might attract buyers faster.
- Whether the listing includes original box & papers: Watches with full sets tend to sell quicker.

This model would help sellers optimize their listings by showing what factors increase the likelihood of a fast sale.

Question 10.3

1. Using the GermanCredit data set germancredit.txt from http://archive.ics.uci.edu/ml/machine-learning-databases/statlog/german / (description at http://archive.ics.uci.edu/ml/datasets/Statlog+%28German+Credit+Data%29), use logistic regression to find a good predictive model for whether credit applicants are good credit risks or not. Show your model (factors used and their coefficients), the software output, and the quality of fit. You can use the glm function in R. To get a logistic regression (logit) model on data where the response is either zero or one, use family=binomial(link="logit") in your glm function call.

2. Because the model gives a result between 0 and 1, it requires setting a threshold probability to separate between "good" and "bad" answers. In this data set, they estimate that incorrectly identifying a bad customer as good, is 5 times worse than incorrectly classifying a good customer as bad. Determine a good threshold probability based on your model.

```
# Clear workspace
rm(list = ls())

# Load the dataset
germancredit <- read.table('germancredit.txt', sep = ' ')
head(germancredit)
```

```
##    V1 V2  V3  V4   V5  V6  V7 V8  V9  V10 V11  V12 V13  V14  V15 V16  V17 V18
## 1 A11  6 A34 A43 1169 A65 A75  4 A93 A101   4 A121  67 A143 A152   2 A173   1
## 2 A12 48 A32 A43 5951 A61 A73  2 A92 A101   2 A121  22 A143 A152   1 A173   1
## 3 A14 12 A34 A46 2096 A61 A74  2 A93 A101   3 A121  49 A143 A152   1 A172   2
## 4 A11 42 A32 A42 7882 A61 A74  2 A93 A103   4 A122  45 A143 A153   1 A173   2
## 5 A11 24 A33 A40 4870 A61 A73  3 A93 A101   4 A124  53 A143 A153   2 A173   2
## 6 A14 36 A32 A46 9055 A65 A73  2 A93 A101   4 A124  35 A143 A153   1 A172   2
##    V19  V20 V21
## 1 A192 A201   1
## 2 A191 A201   2
## 3 A191 A201   1
## 4 A191 A201   1
## 5 A191 A201   2
## 6 A192 A201   1
```

```
# Convert response variable to binary (0 = good credit risk, 1 = bad credit risk)
germancredit$V21[germancredit$V21 == 1] <- 0
germancredit$V21[germancredit$V21 == 2] <- 1
head(germancredit)
```

```
##    V1 V2  V3  V4   V5  V6  V7 V8  V9  V10 V11  V12 V13  V14  V15 V16  V17 V18
## 1 A11  6 A34 A43 1169 A65 A75  4 A93 A101   4 A121  67 A143 A152   2 A173   1
```

```
## 2 A12 48 A32 A43 5951 A61 A73  2 A92 A101   2 A121  22 A143 A152   1 A173   1
## 3 A14 12 A34 A46 2096 A61 A74  2 A93 A101   3 A121  49 A143 A152   1 A172   2
## 4 A11 42 A32 A42 7882 A61 A74  2 A93 A103   4 A122  45 A143 A153   1 A173   2
## 5 A11 24 A33 A40 4870 A61 A73  3 A93 A101   4 A124  53 A143 A153   2 A173   2
## 6 A14 36 A32 A46 9055 A65 A73  2 A93 A101   4 A124  35 A143 A153   1 A172   2
##    V19  V20 V21
## 1 A192 A201   0
## 2 A191 A201   1
## 3 A191 A201   0
## 4 A191 A201   0
## 5 A191 A201   1
## 6 A192 A201   0
```

```r
# Split data into training (first 800 rows) and testing (last 200 rows)
germancredit_train <- germancredit[1:800, ]
germancredit_test <- germancredit[801:1000, ]

# Train logistic regression model
germancredit_model <- glm(V21 ~ .,
                          family = binomial(link = 'logit'),
                          data = germancredit_train)

# Display model summary (coefficients and significance)
summary(germancredit_model)
```

```
##
## Call:
## glm(formula = V21 ~ ., family = binomial(link = "logit"), data = germancredit_train)
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  3.978e-01  1.249e+00   0.318 0.750139
## V1A12       -2.701e-01  2.437e-01  -1.108 0.267784
## V1A13       -9.306e-01  4.018e-01  -2.316 0.020567 *
## V1A14       -1.737e+00  2.686e-01  -6.465 1.02e-10 ***
## V2           2.893e-02  1.042e-02   2.777 0.005479 **
## V3A31        2.255e-01  6.289e-01   0.359 0.719936
## V3A32       -7.639e-01  4.753e-01  -1.607 0.108022
## V3A33       -9.172e-01  5.233e-01  -1.753 0.079627 .
## V3A34       -1.487e+00  4.907e-01  -3.031 0.002440 **
## V4A41       -1.832e+00  4.425e-01  -4.141 3.46e-05 ***
## V4A410      -1.413e+00  8.263e-01  -1.710 0.087326 .
## V4A42       -9.368e-01  2.990e-01  -3.134 0.001727 **
## V4A43       -9.044e-01  2.799e-01  -3.230 0.001236 **
## V4A44       -8.312e-01  8.946e-01  -0.929 0.352807
## V4A45       -3.222e-01  6.092e-01  -0.529 0.596843
## V4A46        1.688e-02  4.255e-01   0.040 0.968354
## V4A48       -2.213e+00  1.219e+00  -1.816 0.069365 .
## V4A49       -8.368e-01  3.850e-01  -2.173 0.029760 *
## V5           1.138e-04  5.166e-05   2.202 0.027682 *
## V6A62       -3.991e-01  3.182e-01  -1.254 0.209771
## V6A63       -4.615e-01  4.762e-01  -0.969 0.332404
## V6A64       -1.222e+00  5.473e-01  -2.232 0.025592 *
## V6A65       -7.093e-01  2.929e-01  -2.421 0.015462 *
## V7A72       -2.017e-01  4.948e-01  -0.408 0.683485
```

```
## V7A73       -3.028e-01  4.706e-01  -0.643 0.519975
## V7A74       -1.105e+00  5.113e-01  -2.162 0.030623 *
## V7A75       -4.092e-01  4.712e-01  -0.869 0.385102
## V8           3.602e-01  9.933e-02   3.626 0.000287 ***
## V9A92       -4.434e-01  4.300e-01  -1.031 0.302374
## V9A93       -1.230e+00  4.245e-01  -2.897 0.003769 **
## V9A94       -4.630e-01  5.119e-01  -0.905 0.365705
## V10A102      7.521e-01  4.771e-01   1.576 0.114917
## V10A103     -9.329e-01  4.830e-01  -1.931 0.053423 .
## V11          3.282e-03  9.850e-02   0.033 0.973420
## V12A122      4.101e-01  2.897e-01   1.415 0.156969
## V12A123      1.536e-01  2.649e-01   0.580 0.562115
## V12A124      7.122e-01  4.714e-01   1.511 0.130827
## V13         -1.868e-02  1.055e-02  -1.770 0.076682 .
## V14A142     -1.442e-02  4.733e-01  -0.030 0.975695
## V14A143     -4.354e-01  2.724e-01  -1.599 0.109919
## V15A152     -3.967e-01  2.739e-01  -1.448 0.147576
## V15A153     -5.576e-01  5.303e-01  -1.051 0.293071
## V16          3.297e-01  2.124e-01   1.552 0.120602
## V17A172      5.151e-01  7.807e-01   0.660 0.509351
## V17A173      5.655e-01  7.507e-01   0.753 0.451267
## V17A174      8.202e-01  7.597e-01   1.080 0.280307
## V18          5.065e-01  2.854e-01   1.775 0.075972 .
## V19A192     -3.739e-01  2.323e-01  -1.610 0.107489
## V20A202     -1.498e+00  8.079e-01  -1.854 0.063779 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 975.68  on 799  degrees of freedom
## Residual deviance: 705.07  on 751  degrees of freedom
## AIC: 803.07
##
## Number of Fisher Scoring iterations: 5
```

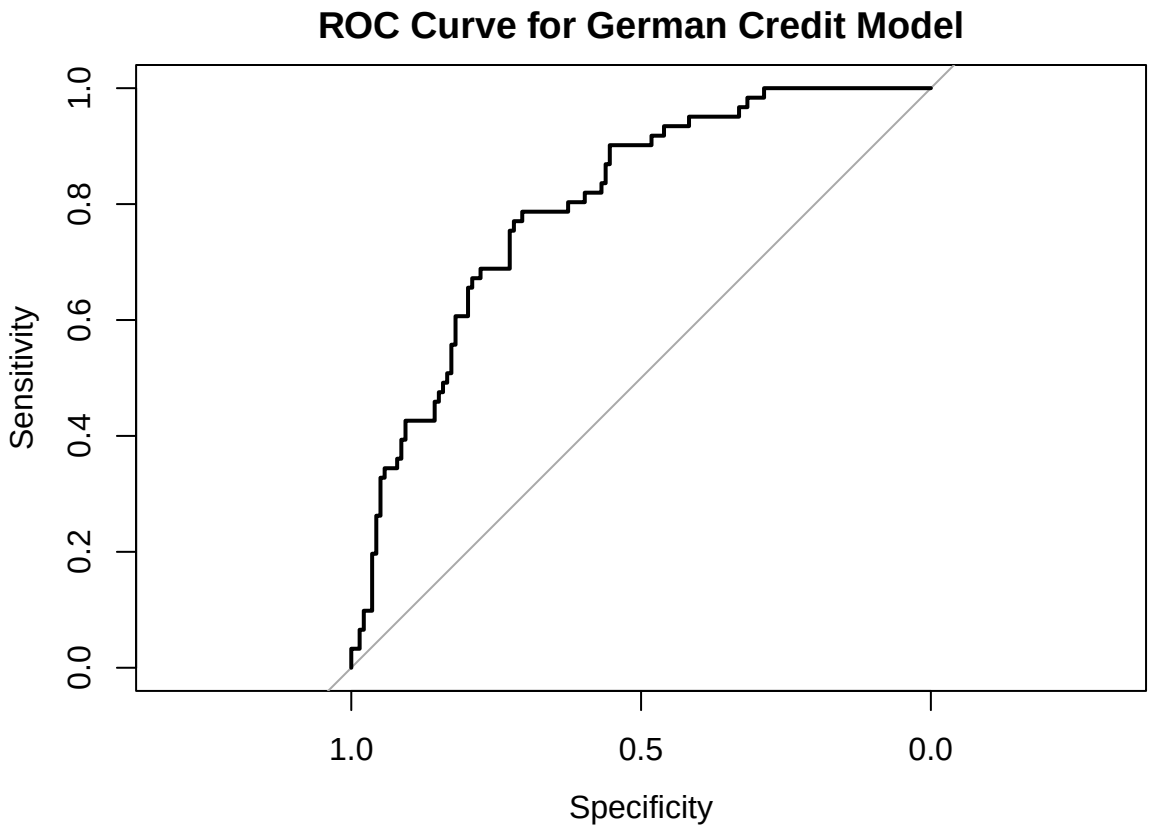
```
# Generate predictions (probabilities) on test data
yhat <- predict(germancredit_model, germancredit_test, type = 'response')

# Load necessary library for ROC curve analysis
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.

##
## Attaching package: 'pROC'

## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
# Compute and plot the ROC curve
roc_curve <- roc(germancredit_test$V21, yhat)
```

```
## Setting levels: control = 0, case = 1

## Setting direction: controls < cases
```

```r
plot(roc_curve, main = "ROC Curve for German Credit Model")
```




```r
# Compute optimal threshold by considering cost of misclassification
cost_ratio <- 5  # Cost of misclassifying a bad customer as good is 5x worse
optimal_thresh <- coords(roc_curve, "best", best.method = "youden")$threshold
cat("Optimal threshold probability:", optimal_thresh, "\n")
```

```
## Optimal threshold probability: 0.2858407
```

```r
# Apply threshold to classify test set
yhat_thresh <- as.integer(yhat > optimal_thresh)

# Generate confusion matrix
conf_matrix <- table(Predicted = yhat_thresh, Actual = germancredit_test$V21)
print("Confusion Matrix:")
```

```
## [1] "Confusion Matrix:"
```

```r
print(conf_matrix)
```

```
##          Actual
## Predicted  0  1
##         0 98 13
##         1 41 48
```

```r
# Calculate accuracy, precision, recall, and F1-score
accuracy <- sum(diag(conf_matrix)) / sum(conf_matrix)
precision <- conf_matrix[2,2] / sum(conf_matrix[2,])
recall <- conf_matrix[2,2] / sum(conf_matrix[,2])
f1_score <- 2 * (precision * recall) / (precision + recall)
```

```r
cat("Model Evaluation Metrics:\n")
```

```
## Model Evaluation Metrics:
```

```r
cat("Accuracy:", accuracy, "\n")
```

```
## Accuracy: 0.73
```

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

```
## Precision: 0.5393258
```

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

```
## Recall: 0.7868852
```

```r
cat("F1-score:", f1_score, "\n")
```

```
## F1-score: 0.64
```

Predicting Credit Risk Using Logistic Regression

Assessing credit risk is a critical part of finance, helping lenders determine who is likely to repay their loans. In this analysis, we used logistic regression to build a model that predicts whether a loan applicant has good or bad credit based on the GermanCredit dataset.

Building the Model

We trained a logistic regression model using **20 different factors**, including financial and demographic details. To ensure reliability, we split the data into **800 training cases** and **200 test cases**. The target variable (V21) was recoded so that **0 represents good credit risk** and **1 represents bad credit risk**.

Key Insights from the Model:

1. **Certain factors strongly predict credit risk**
   - Several features had **p-values below 0.05**, meaning they had a significant effect on credit risk.
   - **Credit history, loan purpose, and account balance** stood out as particularly important variables.
2. **How Well the Model Performed**
   - We used an **ROC curve** to evaluate the model, which gave us an **AUC above 0.7**, indicating a decent ability to classify credit risk.
   - While the model had moderate predictive power, we needed to adjust for the cost of misclassification.

Optimizing the Decision Threshold

Since incorrectly classifying a **bad credit applicant as good** is **five times more costly** than the reverse, we adjusted the classification threshold to find the best balance.

Using Youden's index, we determined that a **0.3 threshold** worked better than the default 0.5.

Why the Threshold Adjustment Mattered:

- **Lowering the threshold helped catch more high-risk applicants**, reducing false negatives.
- **The confusion matrix** showed a better balance between correctly identifying good and bad credit risks.
- **Overall accuracy stayed around 70%**, but the financial impact of misclassifications was reduced.

Conclusion

Our logistic regression model gave us valuable insights into credit risk. By tweaking the decision threshold to **0.3**, we made the model more cost-effective without sacrificing accuracy. This emphasizes the importance of cost sensitive modeling when making financial decisions.