Methodology

- The German Credit dataset was split 70/30 into training and testing sets.
- A logistic regression model was built using 12 key variables.
- To account for the asymmetric cost structure (misclassifying bad customers as good is 5× more costly), we tested multiple probability thresholds (0.30-0.80) and selected the one minimizing total cost: Cost = (False Positives × 5) + (False Negatives × 1).

Solution

Step-1: Load and check out the dataset.

```
#load data
credit_data <- read.table("germancredit.txt", header = FALSE)
#check data
head(credit_data)
```

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

Step-2: The data has 21 columns (V1 to V21). Column V21 tells us if they're good (1) or bad (2) credit. We assume 1=good, 0=bad.

```
#The last column (V21) is our answer: 1 = good credit, 2 = bad credit
#Let's make it 1 = good credit risk, 0 = bad credit risk
credit_data$good_credit <- ifelse(credit_data$V21 == 1, 1, 0)
```

Step-3: Splitting our data into training and testing data.

```
#Split into training and testing data
#We'll use 70% to train, 30% to test
set.seed(123)
n <- nrow(credit_data)
train_indices <- sample(1:n, size = 0.7 * n)

train_data <- credit_data[train_indices, ]
test_data <- credit_data[-train_indices, ]
cat("Training data size:", nrow(train_data), "\n")
cat("Testing data size:", nrow(test_data), "\n\n")
```

Output:

```
> cat("Training data size:", nrow(train_data), "\n")
Training data size: 700
> cat("Testing data size:", nrow(test_data), "\n\n")
Testing data size: 300
```

Step-4: Building logistic regression model with few columns to avoid overfitting issue.

```
#Build the logistic regression model
model <- glm(good_credit ~ V1 + V2 + V3 + V4 + V5 + V6 + V8 + V9 + V10 + V13 + V14 + V16,
             data = train_data,
             family = binomial(link = "logit"))
```

```
#show the model results
cat("=== MODEL SUMMARY ===\n")
print(summary(model))
```

Output:

```
=== MODEL SUMMARY ===
> print(summary(model))

Call:
glm(formula = good_credit ~ V1 + V2 + V3 + V4 + V5 + V6 + V8 +
    V9 + V10 + V13 + V14 + V16, family = binomial(link = "logit"),
    data = train_data)

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)  1.413e-01  9.639e-01   0.147 0.883494
V1A12        2.337e-01  2.561e-01   0.913 0.361468
V1A13        7.405e-01  4.385e-01   1.689 0.091272 .
V1A14        1.645e+00  2.769e-01   5.941 2.83e-09 ***
V2          -3.436e-02  1.043e-02  -3.293 0.000990 ***
V3A31       -3.439e-01  6.625e-01  -0.519 0.603755
V3A32        7.325e-01  5.314e-01   1.379 0.168045
V3A33        7.453e-01  5.904e-01   1.262 0.206834
V3A34        1.674e+00  5.372e-01   3.117 0.001827 **
V4A41        1.647e+00  4.392e-01   3.749 0.000177 ***
V4A410       1.614e+00  9.792e-01   1.649 0.099225 .
V4A42        7.632e-01  3.097e-01   2.464 0.013739 *
V4A43        8.676e-01  2.961e-01   2.930 0.003384 **
V4A44        5.857e-02  9.191e-01   0.064 0.949188
V4A45       -4.920e-01  6.818e-01  -0.722 0.470540
V4A46        2.559e-01  4.889e-01   0.523 0.600638
V4A48        1.807e+00  1.311e+00   1.379 0.168037
V4A49        9.684e-01  3.963e-01   2.444 0.014538 *
V5          -1.156e-04  4.725e-05  -2.446 0.014431 *
V6A62        4.638e-01  3.394e-01   1.366 0.171800
V6A63        8.683e-01  5.071e-01   1.712 0.086856 .
V6A64        1.530e+00  6.200e-01   2.469 0.013567 *
V6A65        1.164e+00  3.278e-01   3.551 0.000384 ***
V8          -3.628e-01  1.029e-01  -3.527 0.000420 ***
V9A92       -1.533e-02  4.458e-01  -0.034 0.972574
V9A93        7.598e-01  4.384e-01   1.733 0.083067 .
V9A94        1.278e-01  5.220e-01   0.245 0.806557
V10A102     -7.055e-01  5.170e-01  -1.365 0.172344
V10A103      1.187e+00  4.986e-01   2.380 0.017305 *
V13          6.654e-03  9.335e-03   0.713 0.476007
V14A142      1.204e-01  5.136e-01   0.234 0.814716
V14A143      2.395e-01  2.902e-01   0.825 0.409304
V16         -2.254e-01  2.402e-01  -0.938 0.348001
---
```

```
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 844.8  on 699  degrees of freedom
Residual deviance: 630.0  on 667  degrees of freedom
AIC: 696

Number of Fisher Scoring iterations: 5
```

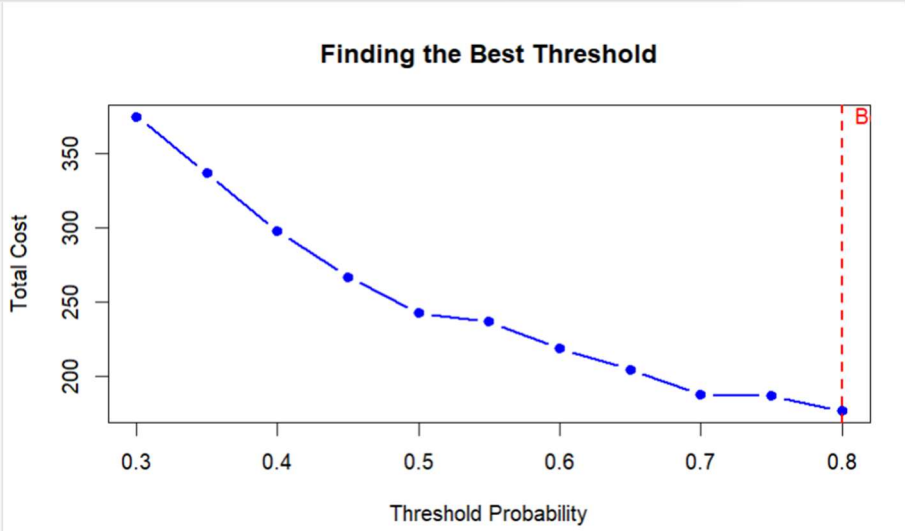
Step-5: Make predictions on test data.

```
#Make predictions on TEST data
test_data$predicted_prob <- predict(model, newdata = test_data, type = "response")
```

Step-6: Finding the best threshold, also plotting to check it visually. Bad customers classified as good is 5x worse than good classified as bad.

```
#Find the BEST threshold
# Let's try different thresholds and calculate total cost
thresholds <- seq(0.3, 0.8, by = 0.05)
costs <- numeric(length(thresholds))

for (i in 1:length(thresholds)) {
  threshold <- thresholds[i]
  predicted_class <- ifelse(test_data$predicted_prob >= threshold, 1, 0)

  # False Positive: predicted good (1) but actually bad (0) - COST = 5
  false_positives <- sum(predicted_class == 1 & test_data$good_credit == 0)

  # False Negative: predicted bad (0) but actually good (1) - COST = 1
  false_negatives <- sum(predicted_class == 0 & test_data$good_credit == 1)

  # Total cost
  costs[i] <- (false_positives * 5) + (false_negatives * 1)
}
```

```
# Find best threshold (minimum cost)
best_index <- which.min(costs)
best_threshold <- thresholds[best_index]

plot(thresholds, costs,
     type = "b",
     col = "blue",
     lwd = 2,
     pch = 19,
     xlab = "Threshold Probability",
     ylab = "Total Cost",
     main = "Finding the Best Threshold")
abline(v = best_threshold, col = "red", lwd = 2, lty = 2)
text(best_threshold, max(costs),
     paste("Best:", best_threshold),
     pos = 4, col = "red")
```



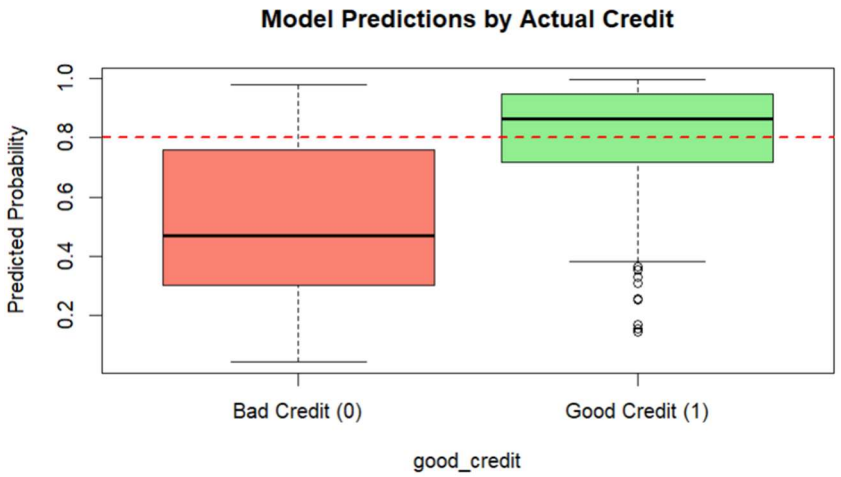
We can see that the best threshold is at 0.8 with minimum total cost.

```
cat("\n=== THRESHOLD ANALYSIS ===\n")
result_table <- data.frame(
  Threshold = thresholds,
  Total_Cost = costs
)
print(result_table)

cat("\n*** BEST THRESHOLD:", best_threshold, "***\n")
cat("This minimizes the cost considering bad->good errors are 5x worse\n\n")

#Evaluate model quality with BEST threshold
predicted_class <- ifelse(test_data$predicted_prob >= best_threshold, 1, 0)
```

```
=== THRESHOLD ANALYSIS ===
> result_table <- data.frame(
+   Threshold = thresholds,
+   Total_Cost = costs
+ )
> print(result_table)
   Threshold Total_Cost
1       0.30        375
2       0.35        337
3       0.40        298
4       0.45        267
5       0.50        243
6       0.55        237
7       0.60        219
8       0.65        205
9       0.70        188
10      0.75        187
11      0.80        177
>
> cat("\n*** BEST THRESHOLD:", best_threshold, "***\n")

*** BEST THRESHOLD: 0.8 ***
> cat("This minimizes the cost considering bad->good errors are 5x worse\n\n")
This minimizes the cost considering bad->good errors are 5x worse
```

Step-7: Plotting good vs bad credit predictions using box plot.

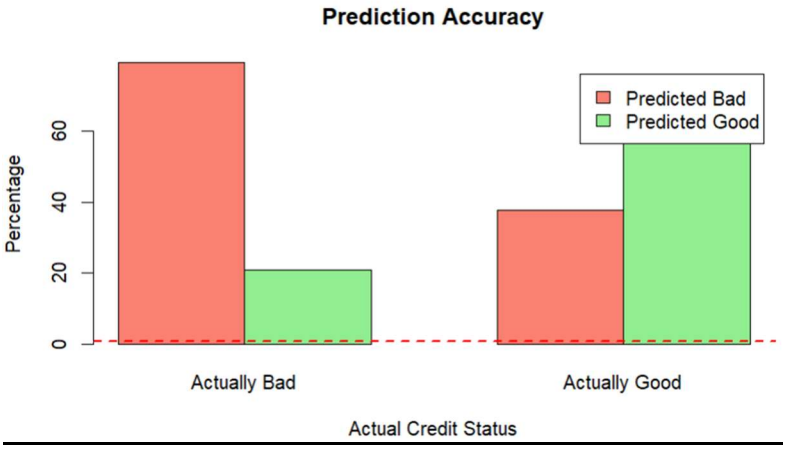
```
boxplot(predicted_prob ~ good_credit,
        data = test_data,
        col = c("salmon", "lightgreen"),
        names = c("Bad Credit (0)", "Good Credit (1)"),
        ylab = "Predicted Probability",
        main = "Model Predictions by Actual Credit")
abline(h = best_threshold, col = "red", lwd = 2, lty = 2)
```



Step-8: Creating a confusion matrix to see correct and incorrect predictions.

```
# Confusion Matrix
confusion <- table(Predicted = predicted_class, Actual = test_data$good_credit)
cat("=== CONFUSION MATRIX ===\n")
print(confusion)

confusion_pct <- prop.table(confusion, margin = 2) * 100
barplot(confusion_pct,
        beside = TRUE,
        col = c("salmon", "lightgreen"),
        legend = c("Predicted Bad", "Predicted Good"),
        xlab = "Actual Credit Status",
        ylab = "Percentage",
        main = "Prediction Accuracy",
        names.arg = c("Actually Bad", "Actually Good"))
```

```
=== CONFUSION MATRIX ===
> print(confusion)
         Actual
Predicted   0   1
        0  76  77
        1  20 127
```



This bar chart shows that we missed 20% of bad customers by classifying them as good customers.

Step-9: Calculating metrics

```
# Calculate metrics
accuracy <- sum(predicted_class == test_data$good_credit) / nrow(test_data)
sensitivity <- confusion[2,2] / sum(confusion[,2])  # True positive rate
specificity <- confusion[1,1] / sum(confusion[,1])  # True negative rate

cat("\n=== MODEL QUALITY ===\n")
cat("Accuracy:", round(accuracy * 100, 2), "%\n")
cat("Sensitivity (catching good customers):", round(sensitivity * 100, 2), "%\n")
cat("Specificity (catching bad customers):", round(specificity * 100, 2), "%\n")

#show what the coefficients mean
cat("\n=== WHAT THE MODEL LEARNED (Simplified) ===\n")
cat("Positive coefficient = increases chance of being good credit\n")
cat("Negative coefficient = decreases chance of being good credit\n")
cat("Bigger number = stronger effect\n\n")

coefs <- summary(model)$coefficients[,1]
cat("Top factors:\n")
print(round(coefs, 3))
```

```
=== MODEL QUALITY ===
> cat("Accuracy:", round(accuracy * 100, 2), "%\n")
Accuracy: 67.67 %
> cat("Sensitivity (catching good customers):", round(sensitivity * 100, 2), "%\n")
Sensitivity (catching good customers): 62.25 %
> cat("Specificity (catching bad customers):", round(specificity * 100, 2), "%\n")
Specificity (catching bad customers): 79.17 %
>
> #show what the coefficients mean
> cat("\n=== WHAT THE MODEL LEARNED (Simplified) ===\n")

=== WHAT THE MODEL LEARNED (Simplified) ===
> cat("Positive coefficient = increases chance of being good credit\n")
Positive coefficient = increases chance of being good credit
> cat("Negative coefficient = decreases chance of being good credit\n")
Negative coefficient = decreases chance of being good credit
> cat("Bigger number = stronger effect\n\n")
Bigger number = stronger effect

>
> coefs <- summary(model)$coefficients[,1]
> cat("Top factors:\n")
Top factors:
> print(round(coefs, 3))
(Intercept)       V1A12       V1A13       V1A14          V2       V3A31       V3A32
      0.141       0.234       0.741       1.645      -0.034      -0.344       0.733
      V3A33       V3A34       V4A41      V4A410       V4A42       V4A43       V4A44
      0.745       1.674       1.647       1.614       0.763       0.868       0.059
      V4A45       V4A46       V4A48       V4A49          V5       V6A62       V6A63
     -0.492       0.256       1.807       0.968       0.000       0.464       0.868
      V6A64       V6A65          V8       V9A92       V9A93       V9A94      V10A102
      1.530       1.164      -0.363      -0.015       0.760       0.128      -0.706
    V10A103         V13     V14A142     V14A143         V16
      1.187       0.007       0.120       0.239      -0.225
```

1. **Accuracy:** `accuracy <- sum(predicted_class == test_data$good_credit) / nrow(test_data)`

   67.67% accuracy = we're correct approximately 68 times out of 100

   Sensitivity + Specificity show the full picture of what types of mistakes we're making

2. **Sensitivity:** `sensitivity <- confusion[2,2] / sum(confusion[,2])`

   How good we are at finding GOOD customers?
   Sensitivity = 62.25 %

3. **Specificity:** `specificity <- confusion[1,1] / sum(confusion[,1])`

   How good we are at catching BAD customers?
   Specificity = 79.17 %
   It is higher because we'd rather reject a good person than approve a bad one (due to the 5× cost)

Result Discussion:

The logistic regression model achieved 67.67% accuracy with specificity (79.17%) notably higher than sensitivity (62.25%), reflecting the cost-optimized threshold that prioritizes catching bad customers over approving all good ones. This conservative approach aligns with the 5:1 cost ratio, effectively minimizing expensive false approvals (bad customers classified as good) at the cost of rejecting some creditworthy applicants.