Problem 3

2025-09-24

Problem 3: Logistic Regression Redux

```
# NOTE: Data files must be in a 'Data' subdirectory relative to this R Markdown file
# Expected structure:
#   - prob4.Rmd
#   - Data/
#     |- laptop_train.csv
#     |- laptop_test.csv
setwd(dirname(rstudioapi::getSourceEditorContext()$path))

# Read CSV files using relative paths
df_orig <- read_csv("../customerchurn.csv")
```

```
## Rows: 7032 Columns: 7
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (3): PaymentMethod, InternetService, Contract
## dbl (4): Churn, MonthlyCharges, SeniorCitizen, tenure
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

Question (a)

```
str(df_orig) # structure (variable types, first few entries)
```

```
## spc_tbl_ [7,032 x 7] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ Churn          : num [1:7032] 0 0 1 0 1 1 0 0 1 0 ...
##  $ MonthlyCharges : num [1:7032] 29.9 57 53.9 42.3 70.7 ...
##  $ SeniorCitizen  : num [1:7032] 0 0 0 0 0 0 0 0 0 0 ...
##  $ PaymentMethod  : chr [1:7032] "Electronic check" "Mailed check" "Mailed check" "Bank transfer" ..
##  $ InternetService: chr [1:7032] "DSL" "DSL" "DSL" "DSL" ...
##  $ tenure         : num [1:7032] 1 34 2 45 2 8 22 10 28 62 ...
##  $ Contract       : chr [1:7032] "Month-to-month" "One year" "Month-to-month" "One year" ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   Churn = col_double(),
##   ..   MonthlyCharges = col_double(),
##   ..   SeniorCitizen = col_double(),
```

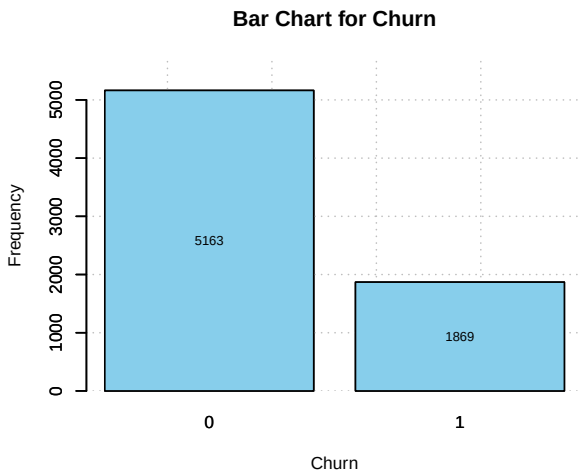
```
##   ..   PaymentMethod = col_character(),
##   ..   InternetService = col_character(),
##   ..   tenure = col_double(),
##   ..   Contract = col_character()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

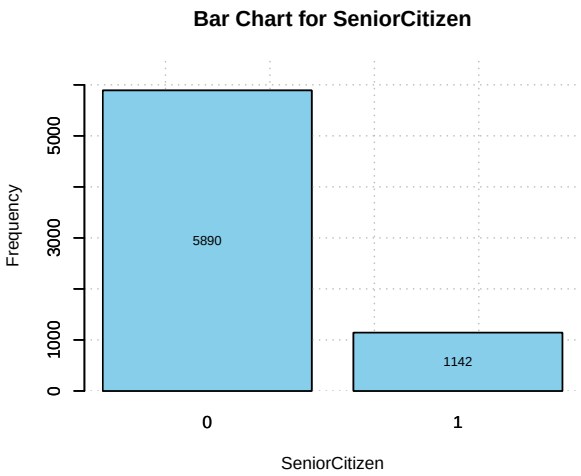
```
summary(df_orig) # summary statistics by column
```

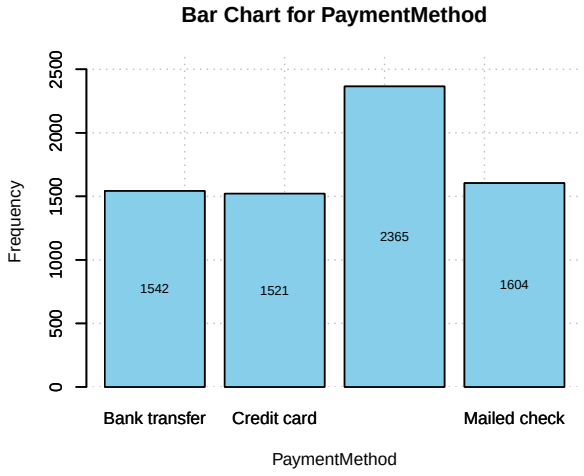
```
##      Churn         MonthlyCharges   SeniorCitizen    PaymentMethod
##  Min.   :0.0000   Min.   : 18.25   Min.   :0.0000   Length:7032
##  1st Qu.:0.0000   1st Qu.: 35.59   1st Qu.:0.0000   Class :character
##  Median :0.0000   Median : 70.35   Median :0.0000   Mode  :character
##  Mean   :0.2658   Mean   : 64.80   Mean   :0.1624
##  3rd Qu.:1.0000   3rd Qu.: 89.86   3rd Qu.:0.0000
##  Max.   :1.0000   Max.   :118.75   Max.   :1.0000
##  InternetService        tenure        Contract
##  Length:7032        Min.   : 1.00   Length:7032
##  Class :character   1st Qu.: 9.00   Class :character
##  Mode  :character   Median :29.00   Mode  :character
##                     Mean   :32.42
##                     3rd Qu.:55.00
##                     Max.   :72.00
```

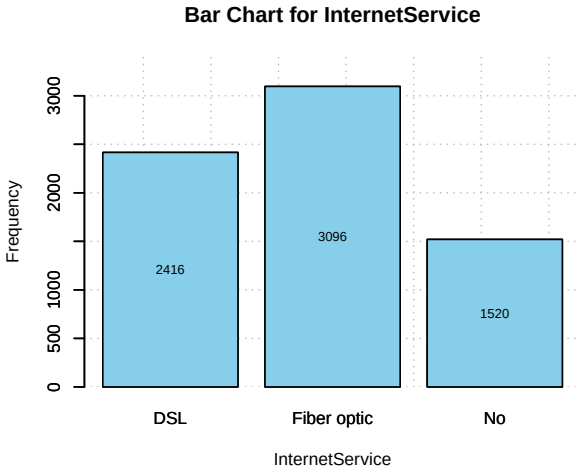
```
# Set up 3x2 grid layout
par(mfrow = c(3, 2))

barchartvector = c("Churn", "SeniorCitizen",
                   "PaymentMethod", "InternetService", "Contract")
for (col_name in barchartvector) {
  counts <- table(df_orig[[col_name]])

  # Create the barplot and store the bar positions
  bar_positions <- barplot(counts,
          main = paste("Bar Chart for", col_name),
          xlab = col_name,
          ylab = "Frequency",
          col = "skyblue",
          ylim = c(0, max(counts) * 1.1))  # Add some space at top

  # Add grid
  grid(nx = NULL, ny = NULL, col = "gray", lty = "dotted", lwd = 1)

  # Redraw the bars on top of the grid
  barplot(counts,
          col = "skyblue",
          add = TRUE)

  # Add text labels inside each bar
  text(x = bar_positions,
       y = counts / 2,  # Position at middle of each bar
       labels = counts,
       cex = 0.8,  # Text size
```

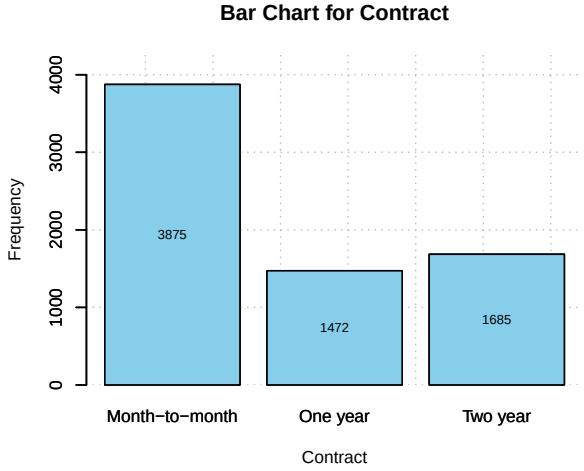
```
        col = "black")  # Text color for visibility
}
```



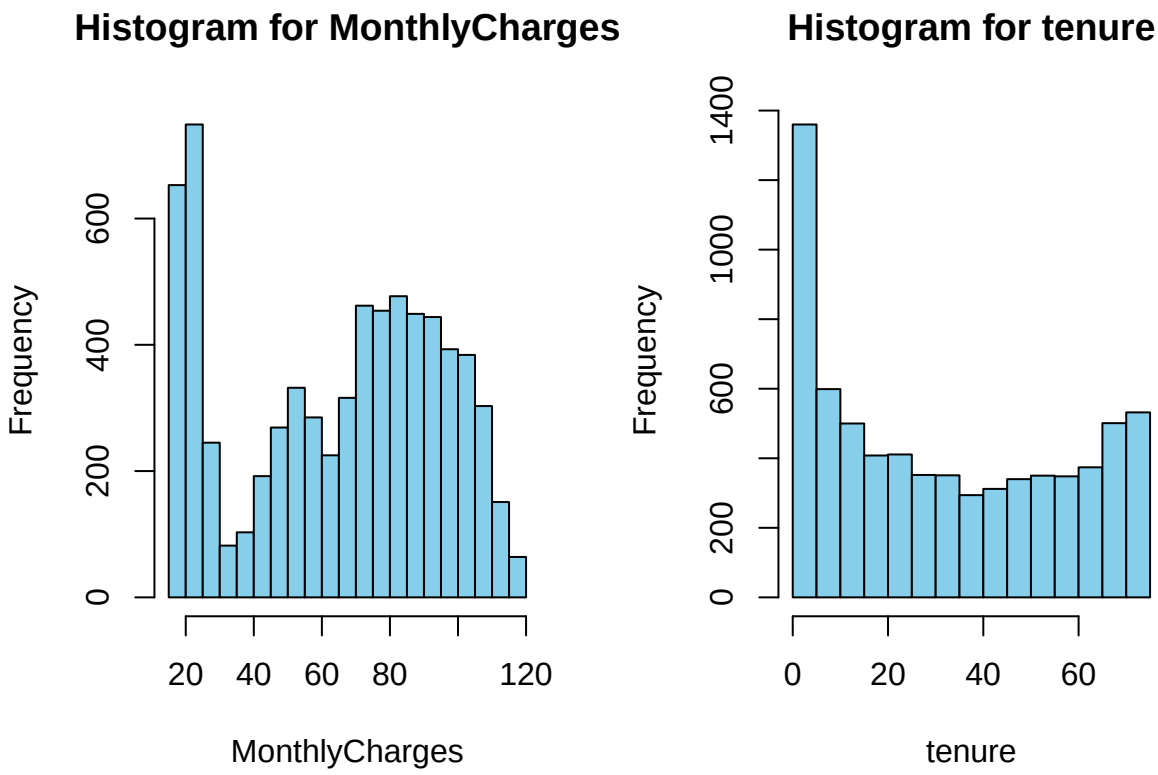








```r
# Set up 1x2 grid layout for histograms
par(mfrow = c(1, 2))

histvector = c("MonthlyCharges", "tenure")
for (col_name in histvector) {
  # Use the actual column values, not table counts
  hist(df_orig[[col_name]],
          main = paste("Histogram for", col_name),
          xlab = col_name,
          ylab = "Frequency",
          col = "skyblue", breaks = "FD")
}
```




```r
# Reset to default single plot layout
par(mfrow = c(1, 1))
```

From this output, we have a few key takeaways: the churn percentage is 26.5785% and the non churn percentage is 73.4215%, meaning we have approx 3x more observations of non churners than churners. We also realize that our data set is also unbalanced in terms of age demographics, having only 16.24% senior citizens. For the payment methods, users have the choice between 4 different payment methods including electronic check which is the most popular choice, and bank transfer/credit card/mailed check which all have pretty similar utilization. To provide more color on internet service, we see that the most popular service is fiber optic followed by DSL, but many people report not having a service. Moreover, the majority of people pay for these services month to month rather than being locked in for 1 or 2 years. With these services, we see that users pay on average of $64.80 per month, with a high of $118.75 and a low of $18.25. Charges are skewed right as is the histogram of tenure.

```r
# Use table() function to see the distribution of the target variable
churn_table <- table(df_orig$Churn)
print(churn_table)
```

```
##
##    0    1
## 5163 1869
```

```r
# Number of customers who did not churn (Churn = 0)
customers_not_churned <- churn_table[1]
cat("Customers who did not churn:", customers_not_churned, "\n")
```

```
## Customers who did not churn: 5163
```

```r
# Number of customers who churned (Churn = 1)
customers_churned <- churn_table[2]
cat("Customers who churned:", customers_churned, "\n")
```

```
## Customers who churned: 1869
```

```r
# Compute churn rate (proportion of customers who churned)
churn_rate <- customers_churned / sum(churn_table)
cat("Churn rate:", round(churn_rate, 4), "or", round(churn_rate * 100, 2), "%\n")
```

```
## Churn rate: 0.2658 or 26.58 %
```

In the above we used table function on the variable Churn to see some characteristics of the variable. We can see that the variable is binary with 0 and 1 values.

Analysis of Customer Churn

From the `table()` function results, we can answer the key questions:

How many customers churned? - **1,869 customers** churned (Churn = 1)

How many customers did not churn? - **5,163 customers** did not churn (Churn = 0)

What is the customers' churn rate? - The churn rate is **26.58%** (or 0.2658 as a proportion) - This means that approximately 1 out of every 4 customers in the dataset churned

Summary: - Total customers: 7,032 (5,163 + 1,869) - Non-churn rate: 73.42% - Churn rate: 26.58%

This indicates a relatively high churn rate, as over a quarter of the customer base has churned. This level of churn represents a significant business concern that warrants further analysis to identify the factors contributing to customer attrition.

[Optional] We can also compute the churn rate using different methods as shown below.

```r
# To compute customer's churn rate
# Method 1: Using mean() function (churn is doubled encoded as 0/1)
churn_rate <- mean(df_orig$Churn == 1)
print(churn_rate)
```

```
## [1] 0.265785
```

Question (b)

```
set.seed(15072)
# creating a binary dependent variable for churn (0 = no, 1 = churn)
df_orig$Churn <- as.factor(df_orig$Churn)

# training (70%) and test (30%) partition
smp_size <- floor(0.70 * nrow(df_orig))
train_ind <- sample(seq_len(nrow(df_orig)), size = smp_size, replace = FALSE)
df_train <- df_orig[train_ind, ]
df_test <- df_orig[-train_ind, ]
```

```
# Fit logistic regression model on training data
mod_logit <- glm(Churn ~ . - PaymentMethod, data = df_train, family = binomial)
summary(mod_logit)
```

```
##
## Call:
## glm(formula = Churn ~ . - PaymentMethod, family = binomial, data = df_train)
##
## Coefficients:
##                              Estimate Std. Error z value Pr(>|z|)
## (Intercept)                -0.492422   0.188257  -2.616   0.0089 **
## MonthlyCharges              0.003523   0.003632   0.970   0.3321
## SeniorCitizen               0.460054   0.095973   4.794 1.64e-06 ***
## InternetServiceFiber optic  1.041354   0.153243   6.795 1.08e-11 ***
## InternetServiceNo          -0.886428   0.177866  -4.984 6.24e-07 ***
## tenure                     -0.033687   0.002497 -13.492  < 2e-16 ***
## ContractOne year           -0.829613   0.124337  -6.672 2.52e-11 ***
## ContractTwo year           -1.736190   0.212439  -8.173 3.02e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 5763.5  on 4921  degrees of freedom
## Residual deviance: 4227.2  on 4914  degrees of freedom
## AIC: 4243.2
##
## Number of Fisher Scoring iterations: 6
```

```
# State the value of and interpret the coefficient for SeniorCitizen
coef_summary <- summary(mod_logit)$coefficients
senior_coef <- coef_summary["SeniorCitizen", "Estimate"]
cat(sprintf("Coefficient for SeniorCitizen: %.4f\n", senior_coef))
```

```
## Coefficient for SeniorCitizen: 0.4601
```

Interpretation:

In the logistic regression model, the `SeniorCitizen` coefficient represents the change in the log-odds of churn for senior citizens compared to non-senior citizens, holding all other variables constant. Since the coefficient is

positive (0.4601), it means that being a senior citizen increases the likelihood of customer churn. Specifically, the odds of churn for senior citizens are $\exp(0.4601) \approx 1.58$ times the odds for non-senior citizens, all else being the same. In other words, senior citizens are about 58% more likely to churn than non-senior citizens, holding all other factors constant.

Question (c)

So essentially I need to find the 5th customer from the initial dataset and predict the probability of churn for that customer using the fitted logistic regression model.

1. Get the 5th customer from the original dataset, keeping only the features used to fit the model

```
customer_5 <- df_orig[5, ]
print(customer_5)
```

```
## # A tibble: 1 x 7
##   Churn MonthlyCharges SeniorCitizen PaymentMethod    InternetService tenure
##   <fct>          <dbl>         <dbl> <chr>            <chr>            <dbl>
## 1 1               70.7             0 Electronic check Fiber optic          2
## # i 1 more variable: Contract <chr>
```

2. Predict the probability of churn for the 5th customer

```
churn_prob <- predict(mod_logit, newdata = customer_5, type = "response")
cat(sprintf("Predicted probability of churn for customer 5: %.4f\n", churn_prob))
```

```
## Predicted probability of churn for customer 5: 0.6749
```

Question (d)

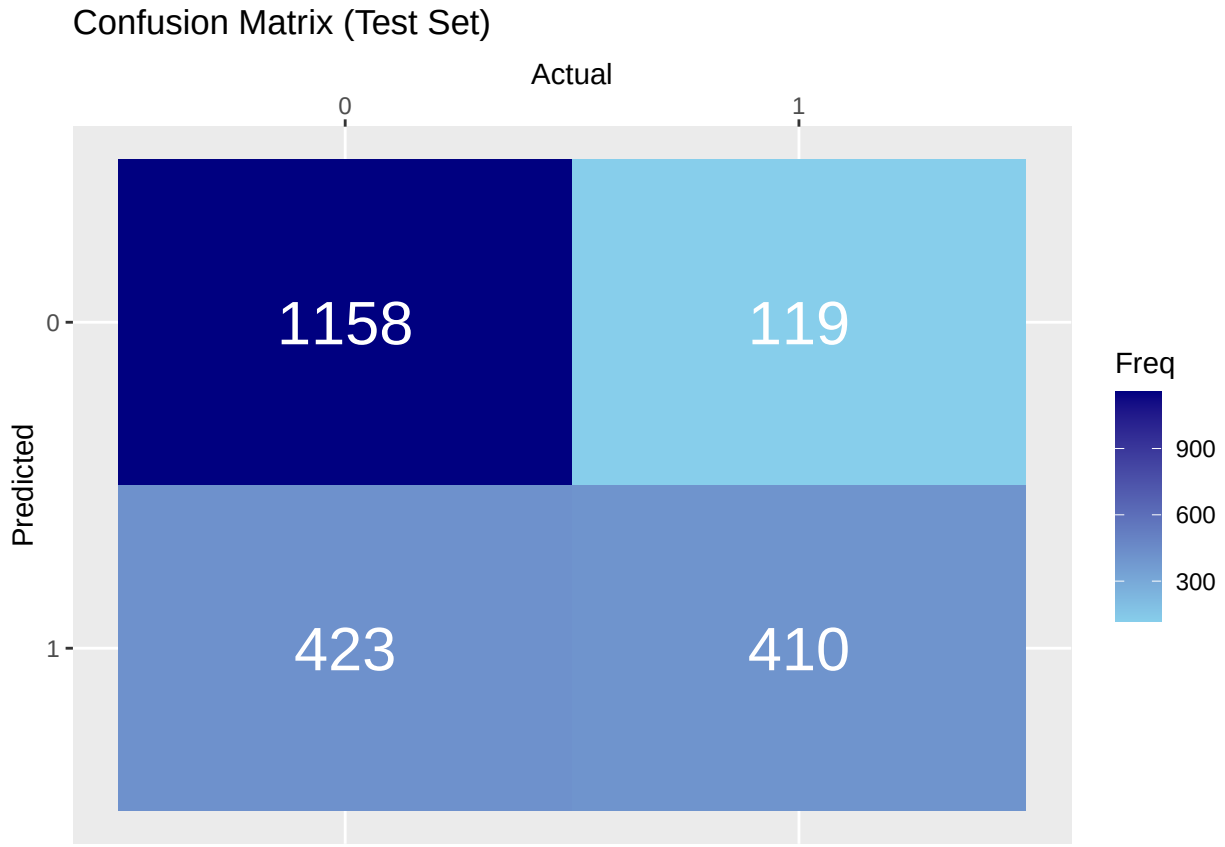
1. Predict probabilities on the test set

```
# Predict probabilities on test set
pred_probs <- predict(mod_logit, newdata = df_test, type = "response")

# Classify based on threshold of 0.3 (any probability higher
# than 30% will be considered as churn)
pred_classes <- ifelse(pred_probs > 0.3, 1, 0)

# Confusion matrix
conf_matrix <- table(Predicted = pred_classes, Actual = df_test$Churn)
print(conf_matrix)
```

```
##          Actual
## Predicted    0    1
##         0 1158  119
##         1  423  410
```

2. Visualize confusion matrix

```r
# Visualize confusion matrix as heatmap
cm_df <- as.data.frame(conf_matrix)
colnames(cm_df) <- c("Predicted", "Actual", "Freq")
cm_df$Predicted <- factor(cm_df$Predicted, levels = c(1, 0))
cm_df$Actual <- factor(cm_df$Actual, levels = c(0, 1))

ggplot(cm_df, aes(x = Actual, y = Predicted, fill = Freq)) +
  geom_tile() +
  geom_text(aes(label = Freq), color = "white", size = 8) +
  scale_fill_gradient(low = "skyblue", high = "navy") +
  labs(
    title = "Confusion Matrix (Test Set)",
    x = "Actual",
    y = "Predicted"
  ) +
  scale_x_discrete(position = "top") # Put Actual labels on top to match table
```



Question (e)

In the confusion matrix, we can identify the following:

- False Positives (FP): These are the cases where the model predicted churn (1) but the actual outcome was no churn (0). In the confusion matrix, this is represented by the cell where Predicted = 1 and Actual = 0. From the confusion matrix, we can see that there are 422 false positives.

- False Negatives (FN): These are the cases where the model predicted no churn (0) but the actual outcome was churn (1). In the confusion matrix, this is represented by the cell where Predicted = 0 and Actual = 1. From the confusion matrix, we can see that there are 119 false negatives.

Question (f)

Here we need to intepret the tradeoff between false positives and false negatives in a managerial context. First of all we need to understand that from a company's perspective, failing to identify a customer who will churn is often more costly than incorrectly predicting that a customer will churn when they won't. This is because if you can identify a customer who is likely to churn, you can take proactive measures to retain them, resulting in a direct financial benefit. On the other hand, if you incorrectly predict that a customer will churn, the cost is usually limited to the resources spent on retention efforts, which may not be as significant as losing a customer.

Now let's analyze the tradeoff from a data perspective: Based on the above, we can understand that if we need to decide between minimizing false positives or false negatives, we should prioritize minimizing false negatives (meaning wrongly predicting a customer will not churn when they actually do).

Therefore, from a managerial perspective, minimizing false negatives (meaning wrongly predicting a customer will not churn when they actually do) is more critical, because missing a true churner means losing revenue and potentially long-term customer value. While false positives incur costs (e.g., retention offers to loyal customers), these costs are generally smaller and can even strengthen customer relationships. As an analyst, I would prioritize a model that does a better job of catching as many real churners as possible, even if it means we sometimes intervene with customers who would have stayed anyway.

Question (g)

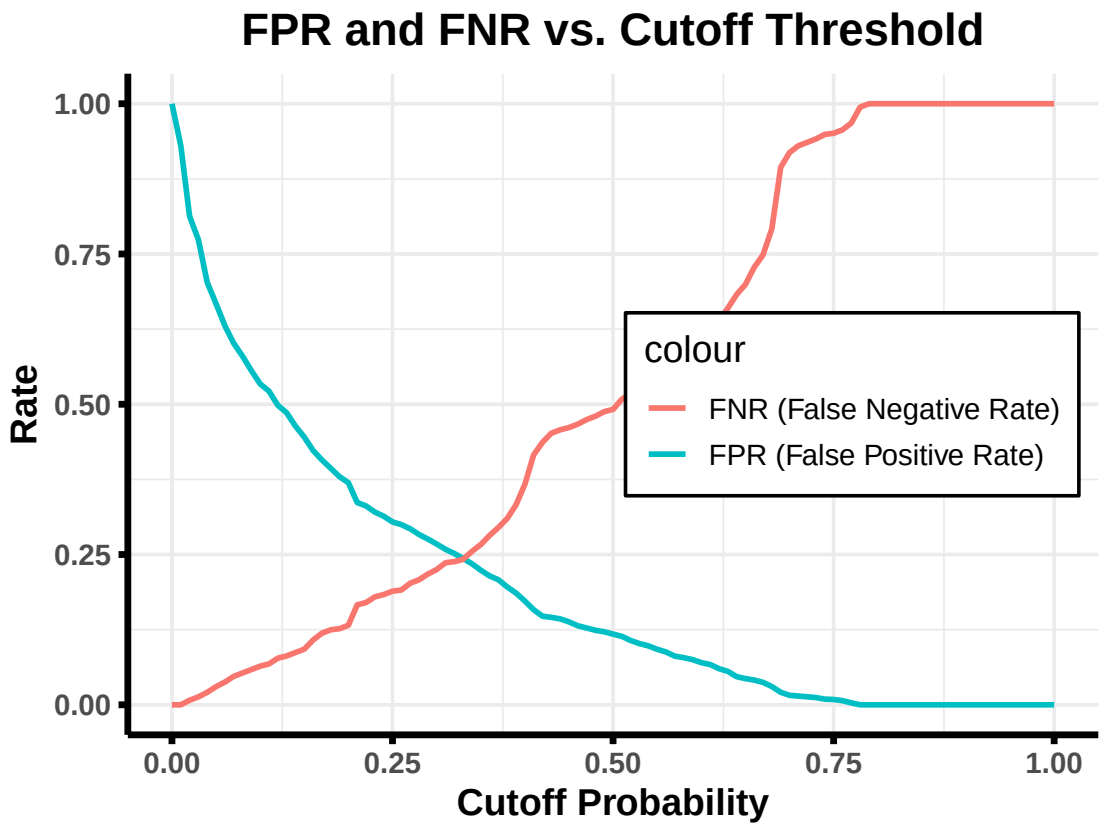
```
# Threshold plot: FPR and FNR vs cutoff

cutoffs <- seq(0, 1, by = 0.01)
fpr <- fnr <- numeric(length(cutoffs))

actual <- df_test$Churn

for (i in seq_along(cutoffs)) {
  pred <- ifelse(pred_probs > cutoffs[i], 1, 0)
  cm <- table(factor(pred, levels = c(0,1)), factor(actual, levels = c(0,1)))
  # cm[1,1]=TN, cm[1,2]=FN, cm[2,1]=FP, cm[2,2]=TP
  fp <- cm[2,1]
  fn <- cm[1,2]
  tn <- cm[1,1]
  tp <- cm[2,2]
  fpr[i] <- ifelse((fp + tn) > 0, fp / (fp + tn), NA)
  fnr[i] <- ifelse((fn + tp) > 0, fn / (fn + tp), NA)
}

threshold_df <- data.frame(
  Cutoff = cutoffs,
  FPR = fpr,
  FNR = fnr
)

ggplot(threshold_df, aes(x = Cutoff)) +
```

```
  geom_line(aes(y = FPR, color = "FPR (False Positive Rate)"), size = 1) +
  geom_line(aes(y = FNR, color = "FNR (False Negative Rate)"), size = 1) +
  labs(
    title = "FPR and FNR vs. Cutoff Threshold",
    x = "Cutoff Probability",
    y = "Rate"
  ) +
  theme_minimal(base_size = 14) +
  theme(
    axis.title = element_text(face = "bold"),
    axis.text = element_text(face = "bold"),
    plot.title = element_text(face = "bold", hjust = 0.5),
    axis.line = element_line(size = 1.2, colour = "black"),
    axis.ticks = element_line(size = 1.2, colour = "black"),
    plot.margin = margin(10, 30, 10, 30),
    legend.position = c(0.98, 0.5),           # right end, middle
    legend.justification = c("right", "center"),
    legend.background = element_rect(fill = "white", color = "black")
  ) +
  coord_cartesian(xlim = c(0, 1), ylim = c(0, 1))
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
## Warning: The `size` argument of `element_line()` is deprecated as of ggplot2 3.4.0.
## i Please use the `linewidth` argument instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
## Warning: A numeric `legend.position` argument in `theme()` was deprecated in ggplot2
## 3.5.0.
## i Please use the `legend.position.inside` argument of `theme()` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```



1. In the plot, the red curve represents the false negative rate (FNR) and the blue curve represents the false positive rate (FPR). As the cutoff probability increases along the x-axis, the blue FPR curve steadily declines, showing that fewer loyal customers are misclassified as churners. At the same time, the red FNR curve rises, showing that more true churners are missed by the model. Both curves are plotted on the same scale, so the crossing point in the middle range makes it easy to see where the balance between the two errors occurs. At the extremes, the tradeoff is clear: at threshold 0 FPR = 1 and FNR = 0 (everyone is flagged as churner), while at threshold 1 FPR = 0 and FNR = 1 (no churners are caught).

2. This visualization makes the tradeoff very clear: increasing the cutoff reduces false positives but increases false negatives. At very low thresholds, the company would waste significant resources by intervening with many customers who would not leave, while at very high thresholds the company would lose real churners and the revenue tied to them. The most effective managerial decision is to choose a threshold in the middle range, where the balance between the two error rates maximizes profitability and aligns best with business goals.

Question (h)

From a managerial perspective, our goal is to set a probability threshold that maximizes overall profit while balancing the trade-off between false positives and false negatives. The profit matrix makes it clear that missing a customer who churns is extremely costly, whereas wrongly predicting churn for a loyal customer has a smaller financial impact. Therefore, we expect the optimal threshold to lean toward being more aggressive in flagging churners, ensuring that fewer at-risk customers are missed. To identify this point, I evaluated total profit across twenty different thresholds.

The given profit matrix is as follows:

	Actual Churn	Actual Non-Churn
Predicted Churn	**$2000**	**-$1000**
Predicted Non-Churn	**-$6000**	**$3000**

We are plotting the expected profit against 20 different probability thresholds ranging from 0 to 1.

```
# Define profit matrix
profit_matrix <- matrix(
  c(2000, -1000, -6000, 3000), # order: TP, FP, FN, TN
  nrow = 2, byrow = TRUE,
  dimnames = list(
    "Predicted" = c("Churn", "NonChurn"),
    "Actual" = c("Churn", "NonChurn")
  )
)

# Thresholds: 20 equally spaced values between 0 and 1
thresholds <- seq(0, 1, length.out = 20)
profits <- numeric(length(thresholds))
TPs <- FPs <- FNs <- TNs <- numeric(length(thresholds))

actual <- df_test$Churn

for (i in seq_along(thresholds)) {
  pred <- ifelse(pred_probs > thresholds[i], 1, 0)
  # Confusion matrix: rows = predicted, cols = actual
  cm <- table(factor(pred, levels = c(1,0)), factor(actual, levels = c(1,0)))
  # cm[1,1]=TP, cm[1,2]=FP, cm[2,1]=FN, cm[2,2]=TN
  TP <- cm[1,1]
  FP <- cm[1,2]
  FN <- cm[2,1]
  TN <- cm[2,2]
  profits[i] <- TP*2000 + FP*(-1000) + FN*(-6000) + TN*3000
  TPs[i] <- TP
  FPs[i] <- FP
  FNs[i] <- FN
  TNs[i] <- TN
  cat(sprintf("Threshold: %.3f | TP: %d | FP: %d | TN: %d | FN: %d | Profit: $%d\n",
              thresholds[i], TP, FP, TN, FN, profits[i]))
}
```

```
## Threshold: 0.000 | TP: 529 | FP: 1581 | TN: 0 | FN: 0 | Profit: $-523000
## Threshold: 0.053 | TP: 511 | FP: 1046 | TN: 535 | FN: 18 | Profit: $1473000
## Threshold: 0.105 | TP: 493 | FP: 834 | TN: 747 | FN: 36 | Profit: $2177000
## Threshold: 0.158 | TP: 473 | FP: 680 | TN: 901 | FN: 56 | Profit: $2633000
## Threshold: 0.211 | TP: 440 | FP: 531 | TN: 1050 | FN: 89 | Profit: $2965000
## Threshold: 0.263 | TP: 426 | FP: 470 | TN: 1111 | FN: 103 | Profit: $3097000
## Threshold: 0.316 | TP: 404 | FP: 404 | TN: 1177 | FN: 125 | Profit: $3185000
## Threshold: 0.368 | TP: 374 | FP: 331 | TN: 1250 | FN: 155 | Profit: $3237000
## Threshold: 0.421 | TP: 297 | FP: 233 | TN: 1348 | FN: 232 | Profit: $3013000
## Threshold: 0.474 | TP: 277 | FP: 198 | TN: 1383 | FN: 252 | Profit: $2993000
## Threshold: 0.526 | TP: 249 | FP: 163 | TN: 1418 | FN: 280 | Profit: $2909000
```

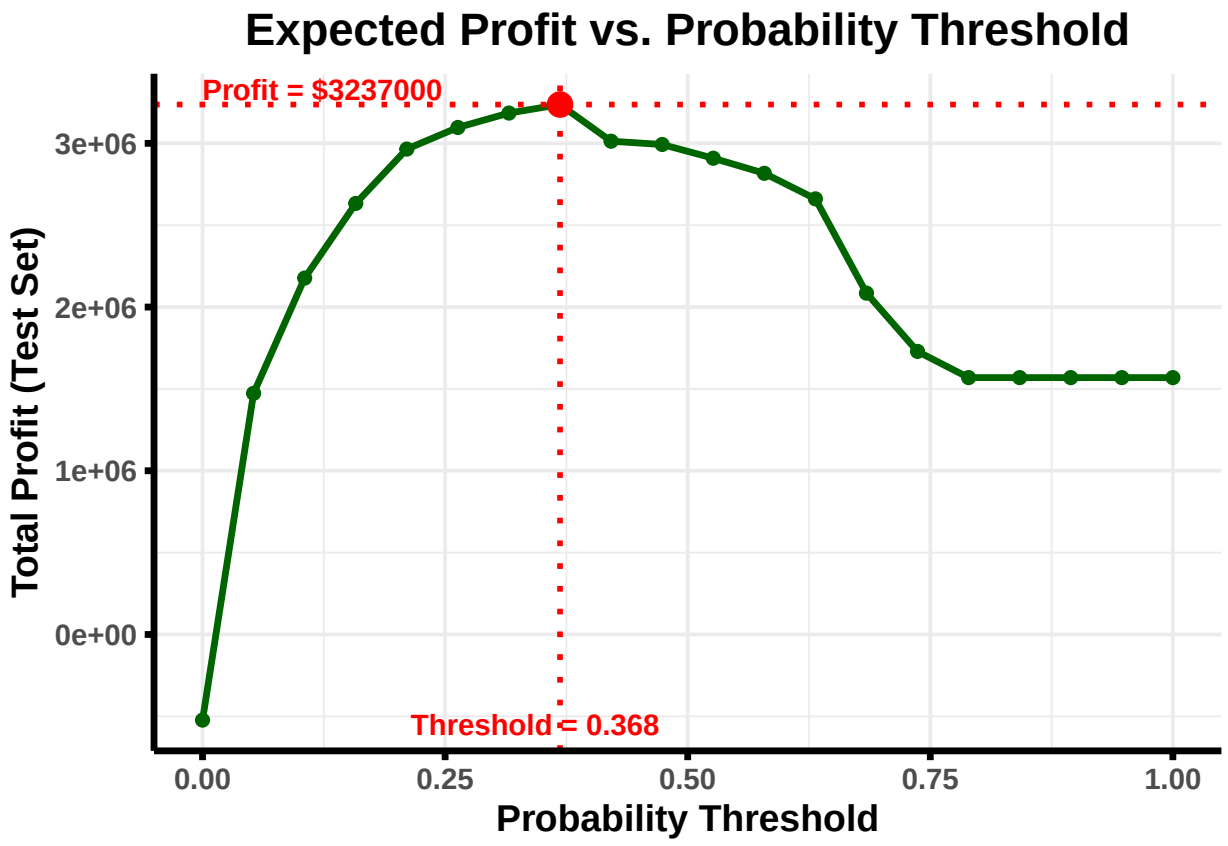
```
## Threshold: 0.579 | TP: 218 | FP: 124 | TN: 1457 | FN: 311 | Profit: $2817000
## Threshold: 0.632 | TP: 180 | FP: 87 | TN: 1494 | FN: 349 | Profit: $2661000
## Threshold: 0.684 | TP: 83 | FP: 37 | TN: 1544 | FN: 446 | Profit: $2085000
## Threshold: 0.737 | TP: 29 | FP: 18 | TN: 1563 | FN: 500 | Profit: $1729000
## Threshold: 0.789 | TP: 0 | FP: 0 | TN: 1581 | FN: 529 | Profit: $1569000
## Threshold: 0.842 | TP: 0 | FP: 0 | TN: 1581 | FN: 529 | Profit: $1569000
## Threshold: 0.895 | TP: 0 | FP: 0 | TN: 1581 | FN: 529 | Profit: $1569000
## Threshold: 0.947 | TP: 0 | FP: 0 | TN: 1581 | FN: 529 | Profit: $1569000
## Threshold: 1.000 | TP: 0 | FP: 0 | TN: 1581 | FN: 529 | Profit: $1569000
```

```r
# Create data frame for plotting
profit_df <- data.frame(
  Threshold = thresholds,
  Profit = profits
)

# Print the best threshold and its stats
best_idx <- which.max(profit_df$Profit)
cat(sprintf("\nBest Threshold: %.3f | TP: %d | FP: %d | TN: %d | FN: %d | Max Profit: $%d\n",
            profit_df$Threshold[best_idx], TPs[best_idx], FPs[best_idx],
            TNs[best_idx], FNs[best_idx], profit_df$Profit[best_idx]))
```

```
##
## Best Threshold: 0.368 | TP: 374 | FP: 331 | TN: 1250 | FN: 155 | Max Profit: $3237000
```

```r
ggplot(profit_df, aes(x = Threshold, y = Profit)) +
  geom_line(color = "darkgreen", size = 1.2) +
  geom_point(color = "darkgreen", size = 2) +
  # Highlight best threshold with a red circle
  geom_point(
    data = profit_df[best_idx, , drop = FALSE],
    aes(x = Threshold, y = Profit),
    color = "red", size = 4, shape = 21, fill = "red"
  ) +
  # Add dotted lines to axes from the best point
  geom_vline(
    xintercept = profit_df$Threshold[best_idx],
    linetype = "dotted", color = "red", size = 1
  ) +
  geom_hline(
    yintercept = profit_df$Profit[best_idx],
    linetype = "dotted", color = "red", size = 1
  ) +
  # Annotate the best point with threshold and profit values
  annotate(
    "text",
    x = profit_df$Threshold[best_idx],
    y = min(profit_df$Profit),
    label = sprintf("Threshold = %.3f", profit_df$Threshold[best_idx]),
    vjust = 0.7, hjust = 0.6, color = "red", fontface = "bold"
  ) +
  annotate(
    "text",
```

```
    x = min(profit_df$Threshold),
    y = profit_df$Profit[best_idx],
    label = sprintf("Profit = $%d", profit_df$Profit[best_idx]),
    vjust = -0.2, hjust = 0, color = "red", fontface = "bold"
  ) +
  labs(
    title = "Expected Profit vs. Probability Threshold",
    x = "Probability Threshold",
    y = "Total Profit (Test Set)"
  ) +
  theme_minimal(base_size = 14) +
  theme(
    axis.title = element_text(face = "bold"),
    axis.text = element_text(face = "bold"),
    axis.line = element_line(size = 1.2, colour = "black"),
    axis.ticks = element_line(size = 1.2, colour = "black"),
    plot.title = element_text(face = "bold", hjust = 0.5)
  )
```



The results confirm this intuition. At very low thresholds, the model predicts nearly everyone as a churner, leading to heavy losses from excessive false positives. As the threshold increases, profits rise steadily, reaching a maximum of $3,237,000 at a threshold of 0.368. At this point, the model correctly identifies 373 churners, with 331 false positives and 155 false negatives. Beyond this threshold, profits start to decline, as the number of missed churners grows and outweighs the savings from fewer false positives.

In the plot, I highlighted this optimal threshold with red dotted lines. It clearly represents the point where

the trade-off between catching churners and avoiding unnecessary retention costs delivers the highest return. From a business standpoint, this suggests adopting a moderately low threshold that focuses on reducing costly missed churners while still managing retention expenses.