Assignment3

Chia-Hao Lo

November 23rd 2024

Setup

```
#Load the required packages
library(tidyverse)
library(tidymodels)
library(modelr)
library(car)
```

Q1. Loading the data

- As a data scientist we need to follow deliverable specifications. So, here we calculate which data set.

```
# Here we use our ysn to identify which data set we need to use
ysn = 1907385
mod4 <- ysn %% 4
mod4
```

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

- Here we get 1 from student number modulo 4, then we use programs_1.csv to be our data set.

```
# Loading the data using the correct tidyverse command
data = read.csv("./data/programs_1.csv")
# Save the data as tibble
data <- as_tibble(data)
# Display the first 10 lines of the data
head(data, 10)
```

```
## # A tibble: 10 x 6
##    FError   LOC XP        foosball Partays B..
##     <int> <int> <chr>     <chr>    <chr>   <chr>
##  1      1 24677 moderate  yes      many    no
##  2      1 27138 extensive no       some    yes
##  3      0 32273 extensive no       some    yes
##  4      0 40079 extensive no       some    yes
##  5      0  8130 moderate  yes      none    yes
##  6      0 10622 moderate  yes      some    no
##  7      1 20048 extensive no       many    no
##  8      1 16032 moderate  yes      some    no
##  9      0 25202 extensive yes      some    yes
## 10      1 25015 minimal   yes      many    yes
```

- The dimension of the data set gives an idea about the size of the data. We get to know how many variables and data points we are going to work with.

```r
dim(data)
```

```
## [1] 36000     6
```

- The dim() function here takes 1 argument which is the data set and gives out the dimension of the program_1.csv data set as rows = 36000, columns = 6

Q2. Identify what types of variables in the current dataset

- – **FError**:
  – **Categorical Nominal**
  – This variable shows whether a fetal error occurred, and it represented by binary values. Since the possible values are 1 or 0 and it represents the meaning of the fatal error occurs or not, thus it is a categorical nominal variable.
- – **LOC**:
  – **Quantitative Discrete**
  – This variable represents the number of lines of code in each program, which is a whole number(e.g.24677, 8130). Since lines of code can't be split into fractions, this is a discrete numeric variable.
- – **XP**:
  – **Categorical Ordinal**
  – XP is represents the debugging team leader's experience level, which categorized as "minimal", "moderate", or "extensive", and three of them have natural order. Thus this is ordinal variable.
- – **foosball**:
  – **Categorical Nominal**
  – This variable means if the office has a foosball table. These categories do not have any natural ordering, only yes or no, thus it is a nominal variable.
- – **Partays**:
  – **Categorical Ordinal**
  – This variable shows the number of office parties attended during the debugging project. With "none", "some", or "many", thus the categories have a natural ranking, so it is a ordinal.
- – **B..**:
  – **Categorical Nominal**
  – This variable indicates if the team received a bonus (yes or no). There is no natural order, which makes it nominal.

Q3. Tame data

Q3.(1). Fix and tame all column names

- From 2. The Data, we know every variable names and tame all names.
  – Ensure all variable names are less than 20 characters.
  – Use **snake_case** (lowercase letters with underscores).
  – Avoid spaces in variable names.

```r
# Rename columns to follow snake case conventions
data <- data %>%
  rename(ferror = FError,
         loc = LOC,
         xp = XP,
         parties = Partays,
         bonus = B..)
```

Q3.(2). Convert the fatal error status to factor type, with yes for 1 and no for 0

```
# Convert the fatal error status
data$ferror <- ifelse(data$ferror == 1, "yes", "no")
# Transfer the variable to factor
data$ferror <- as.factor(data$ferror)
```

Q3.(3). Treat the number of lines of code as a quantitative continuous variable

- For question asked, we convert loc to quantitative continuous because of the predictor should be continuous.

```
# Treat loc as quantitative continuous
data$loc <- as.numeric(data$loc)
```

Q3.(4). Store Categorical Ordinal variables as factor

```
# Here we store xp and parties as factor type, and both of them are also owned natural order
# Convert xp to an ordered factor
data$xp <- factor(data$xp)

#Convert parties to an ordered factor
data$parties <- factor(data$parties)
```

Q3.(5). Conform to the Tame Data conventions

- From **Module 2, page 3**, we tame the data as the rest of the guidelines.
  - Any "yes/no" character encoding of binary variables is converted to "TRUE/FALSE" logical variables.

```
# Ordered Factors, Factors, Characters, and Logicals
data$foosball <- ifelse(data$foosball == "yes", TRUE, FALSE)
data$foosball <- as.logical(data$foosball)
data$bonus <- ifelse(data$bonus == "yes", TRUE, FALSE)
data$bonus <- as.logical(data$bonus)
```

```
# Display the first 10 rows and dimensions
head(data, 10)
```

```
## # A tibble: 10 x 6
##    ferror   loc xp        foosball parties bonus
##    <fct>  <dbl> <fct>     <lgl>    <fct>   <lgl>
##  1 yes    24677 moderate  TRUE     many    FALSE
##  2 yes    27138 extensive FALSE    some    TRUE
##  3 no     32273 extensive FALSE    some    TRUE
##  4 no     40079 extensive FALSE    some    TRUE
##  5 no      8130 moderate  TRUE     none    TRUE
##  6 no     10622 moderate  TRUE     some    FALSE
##  7 yes    20048 extensive FALSE    many    FALSE
##  8 yes    16032 moderate  TRUE     some    FALSE
##  9 no     25202 extensive TRUE     some    TRUE
## 10 yes    25015 minimal   TRUE     many    TRUE
```

```
# Display the dimensions
dim(data)
```

```
## [1] 36000     6
```

Q4. Setting correct seed and split the data

- From Deliverable Specifications, we know that we need to set the seed as our ysn.

```
# Setting the correct seeds
set.seed(ysn)
# Split the data to training and testing dataset (training set with 25,000 rows)
data_split <- initial_split(data, prop = 25/36)
data_training <- training(data_split)
data_testing <- testing(data_split)

# Display train dataset
head(data_training, 10)
```

```
## # A tibble: 10 x 6
##    ferror   loc xp        foosball parties bonus
##    <fct>  <dbl> <fct>     <lgl>    <fct>   <lgl>
##  1 yes    68132 moderate  FALSE    some    FALSE
##  2 no     30666 moderate  FALSE    none    FALSE
##  3 yes    43572 minimal   TRUE     none    FALSE
##  4 yes     7757 minimal   TRUE     many    TRUE
##  5 no     36355 minimal   TRUE     some    FALSE
##  6 no     29126 extensive TRUE     none    TRUE
##  7 no     15231 extensive TRUE     some    TRUE
##  8 yes    13589 moderate  TRUE     many    FALSE
##  9 yes    31098 extensive TRUE     none    TRUE
## 10 no     25954 moderate  TRUE     some    FALSE
```

```
# Display train dataset dimensions
dim(data_training)
```

```
## [1] 25000     6
```

```
# Display test dataset
head(data_testing, 10)
```

```
## # A tibble: 10 x 6
##    ferror   loc xp        foosball parties bonus
##    <fct>  <dbl> <fct>     <lgl>    <fct>   <lgl>
##  1 no      8130 moderate  TRUE     none    TRUE
##  2 yes    30918 moderate  FALSE    none    FALSE
##  3 no     18957 minimal   TRUE     some    FALSE
##  4 yes    10853 minimal   TRUE     none    FALSE
##  5 yes    24284 extensive FALSE    many    FALSE
##  6 yes    17369 minimal   FALSE    some    FALSE
##  7 yes    42692 moderate  TRUE     some    FALSE
##  8 no     17696 extensive TRUE     none    TRUE
##  9 yes    26672 minimal   TRUE     some    TRUE
## 10 no      6490 moderate  TRUE     some    FALSE
```

```
# Display test dataset dimensions
dim(data_testing)
```

```
## [1] 11000     6
```

Q5 Fit a logistic regression model for training data

```
#Getting a logistic regression model with glm engine and fitting it into the training model
lr_spec <- logistic_reg(mode = "classification") %>%
  set_engine("glm")
# Using all variables as predictors without any interaction terms
lr_model <- lr_spec %>%
  fit(ferror ~ ., data = data_training)
# Get the summary from the model
summary(lr_model$fit)
```

```
##
## Call:
## stats::glm(formula = ferror ~ ., family = stats::binomial, data = data)
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -2.186e+00  6.165e-02 -35.468   <2e-16 ***
## loc           6.553e-05  1.147e-06  57.131   <2e-16 ***
## xpminimal     2.319e+00  4.422e-02  52.435   <2e-16 ***
## xpmoderate    1.394e+00  3.979e-02  35.048   <2e-16 ***
## foosballTRUE  2.847e-02  3.102e-02   0.918   0.3588
## partiesnone  -1.369e+00  4.506e-02 -30.390   <2e-16 ***
## partiessome  -1.475e+00  4.178e-02 -35.296   <2e-16 ***
## bonusTRUE    -8.329e-02  3.305e-02  -2.520   0.0117 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 34570  on 24999  degrees of freedom
## Residual deviance: 25917  on 24992  degrees of freedom
## AIC: 25933
##
## Number of Fisher Scoring iterations: 4
```

Q6. How many lines are described from Question 5

- In our model, loc is the only quantitative continuous predictor. Because we will model it linearly, the results in a set of lines will change based on the values of categorical of predictors.
- For the categorical variables, there are xp, foosball, parties, and bonus.
- For xp, there are three variables('minimal', 'moderate', 'extensive') 3 lines
- For foosball, there are two variables('TRUE' or 'FALSE') 2 lines
- For parties, there are three variables('none','some', 'many') 3 lines
- For bonus, there are two variables('TRUE' or 'FALSE') 2 lines

$$Total\ lines: 3 * 2 * 3 * 2 = 36\ lines$$

Q7. Identify all interaction terms that meet the 90% significance level

- From Question 5 we fit our logistic regression model to training dataset, including all individual variables and all second-order interaction terms using .^2

```
lr_model2 <- lr_spec %>%
  fit(ferror ~ .^2, data = data_training)

# Using Anova() to get the p-values for variables and interactions
Anova(lr_model2$fit)
```

```
## Analysis of Deviance Table (Type II tests)
##
## Response: ferror
##                   LR Chisq Df Pr(>Chisq)
## loc                 4198.8  1  < 2.2e-16 ***
## xp                  3260.8  2  < 2.2e-16 ***
## foosball               1.1  1    0.30059
## parties             1448.8  2  < 2.2e-16 ***
## bonus                  6.3  1    0.01199 *
## loc:xp                 6.8  2    0.03362 *
## loc:foosball           0.1  1    0.71245
## loc:parties           50.9  2  9.035e-12 ***
## loc:bonus              0.0  1    0.89813
## xp:foosball            1.5  2    0.48200
## xp:parties             2.0  4    0.74283
## xp:bonus              87.0  2  < 2.2e-16 ***
## foosball:parties       1.8  2    0.41091
## foosball:bonus         0.3  1    0.57932
## parties:bonus          7.3  2    0.02577 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

- After we found the p-value, we need to identify all interaction terms that meet 90% significance level(p-value < 0.1). Below all the interaction terms are meet the 90% significance level:
  - **xp:bonus** : p-value $<$ 2.2e-16 (highly significant)
  - **loc:parties** : p-value = 9.035e-12 (highly significant)
  - **loc:xp : p-value** = 0.03362 (significant)
  - **parties:bonus** : p-value = 0.02577 (significant)

Q8. Backwards stepwise regression

Q8.(a).Ignore foosball table

```
# Here we remove all the interaction with foosball
lr_model3 <- lr_spec %>%
  fit(ferror ~ loc +
        xp +
        parties +
        bonus +
        loc:xp +
        loc:parties +
        loc:bonus +
        xp:parties +
        xp:bonus +
        parties:bonus,
      data = data_training)
# Using Anova() to show the output
```

```r
Anova(lr_model3$fit)
```

```
## Analysis of Deviance Table (Type II tests)
##
## Response: ferror
##               LR Chisq Df Pr(>Chisq)
## loc             4198.3  1  < 2.2e-16 ***
## xp              3260.5  2  < 2.2e-16 ***
## parties         1448.6  2  < 2.2e-16 ***
## bonus              6.3  1    0.01214 *
## loc:xp             6.8  2    0.03354 *
## loc:parties       50.9  2  8.753e-12 ***
## loc:bonus          0.0  1    0.88183
## xp:parties         1.9  4    0.75593
## xp:bonus          86.8  2  < 2.2e-16 ***
## parties:bonus      7.3  2    0.02564 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Q8.(b). Backwards stepwise regression for interaction terms

- From the question, we need to remove the interaction not meet 90% significance level one by one.
- Step 1: We identified loc:bonus has the lowest significance level, which p-value = 0.88183, thus we remove it in this step.

```r
lr_model4 <- lr_spec %>%
  fit(ferror ~ loc +
        xp +
        parties +
        bonus +
        loc:xp +
        loc:parties +
        xp:parties +
        xp:bonus +
        parties:bonus,
      data = data_training)
# Showing the Anova output
Anova(lr_model4$fit)
```

```
## Analysis of Deviance Table (Type II tests)
##
## Response: ferror
##               LR Chisq Df Pr(>Chisq)
## loc             4198.3  1  < 2.2e-16 ***
## xp              3263.2  2  < 2.2e-16 ***
## parties         1448.6  2  < 2.2e-16 ***
## bonus              6.3  1    0.01214 *
## loc:xp             8.4  2    0.01535 *
## loc:parties       50.9  2   8.75e-12 ***
## xp:parties         1.9  4    0.75901
## xp:bonus          93.3  2  < 2.2e-16 ***
## parties:bonus      7.4  2    0.02444 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

- Step 2: After the first step, we found that xp:parties has the lowest significance level, let's remove it.

```
lr_model5 <- lr_spec %>%
  fit(ferror ~ loc +
        xp +
        parties +
        bonus +
        loc:xp +
        loc:parties +
        xp:bonus +
        parties:bonus,
      data = data_training)
# Showing the Anova output
Anova(lr_model5$fit)
```

```
## Analysis of Deviance Table (Type II tests)
##
## Response: ferror
##               LR Chisq Df Pr(>Chisq)
## loc             4198.5  1  < 2.2e-16 ***
## xp              3263.2  2  < 2.2e-16 ***
## parties         1448.6  2  < 2.2e-16 ***
## bonus              6.3  1   0.012373 *
## loc:xp            10.2  2   0.006139 **
## loc:parties       51.0  2  8.634e-12 ***
## xp:bonus          93.0  2  < 2.2e-16 ***
## parties:bonus      6.8  2   0.033446 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

- The final model has no terms should be removed, all interaction terms are all meet 90% significance level now and also following the principle of marginality.

Q8.(c). Backwards stepwise regression for individual terms

- After applying backward stepwise regression, we have reached a model where all individual terms and interaction terms are significant at the 90% level (p-value < 0.10).
  - **loc**: p-value $<$ 2.2e-16 (high significant)
  - **xp**: p-value $<$ 2.2e-16 (high significant)
  - **parties**: p-value $<$ 2.2e-16 (high significant)
  - **bonus**: p-value $= 0.012373$ (significant)
- Since there is no terms below 90% level, no further regression steps are needed.

Q9.

Q9.(a). Interaction terms fitting significance level 90%

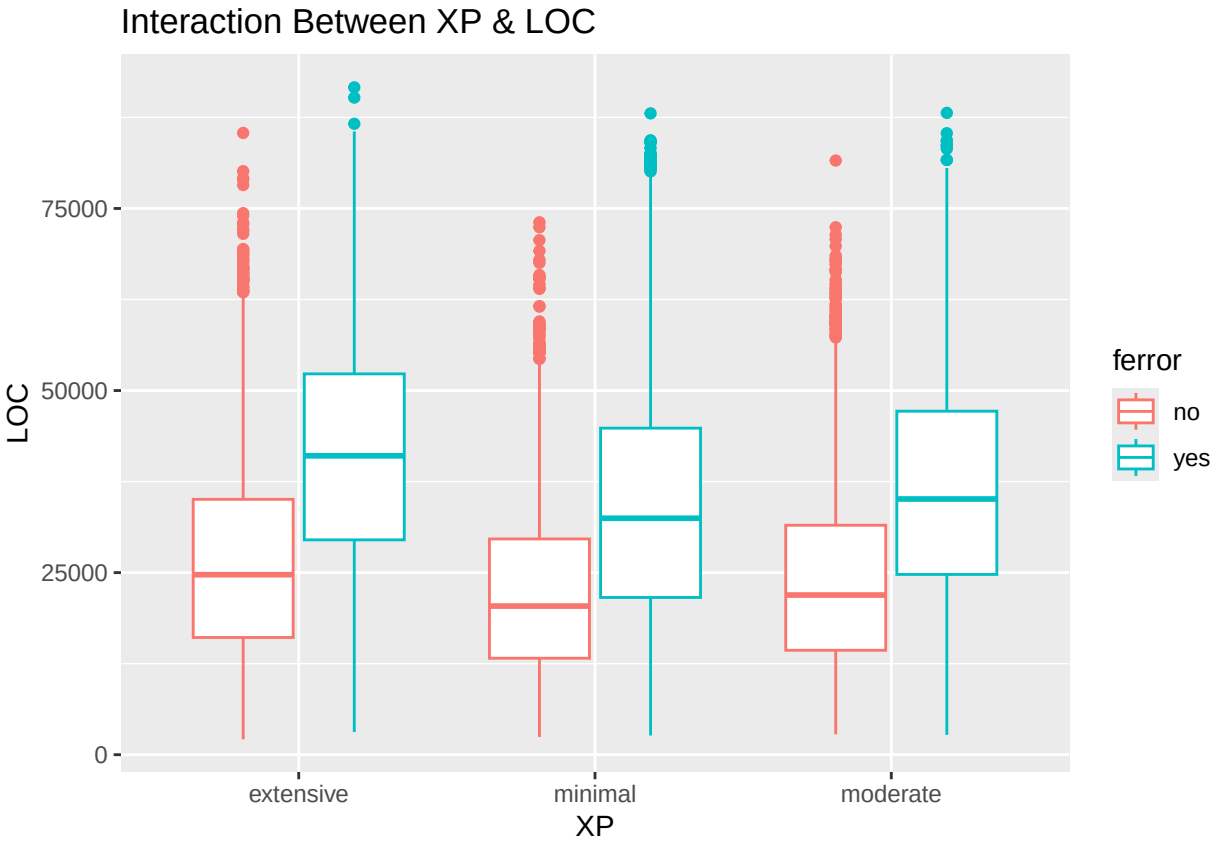
- For the final model, there are 4 interaction terms meet 90% significance level:
  - **loc:xp**: p-value $= 0.006139$ (high significance)
  - **loc:parties**: p-value $=$ 8.634e-12 (high significance)
  - **xp:bonus**: p-value $<$ 2.2e-16 (high significance)
  - **parties:bonus**: p-value $= 0.033446$ (high significance)

Q9.(b). Hypotheses for interaction terms

- For this part, we will use some graphs to prove

First, we plot interaction between xp and loc(xp:loc)

```
ggplot(data_training, aes(x = xp, y = loc, col = ferror)) +
  geom_boxplot() +
  labs(title = "Interaction Between XP & LOC",
       x = "XP",
       y = "LOC",
       fill = "FError")
```

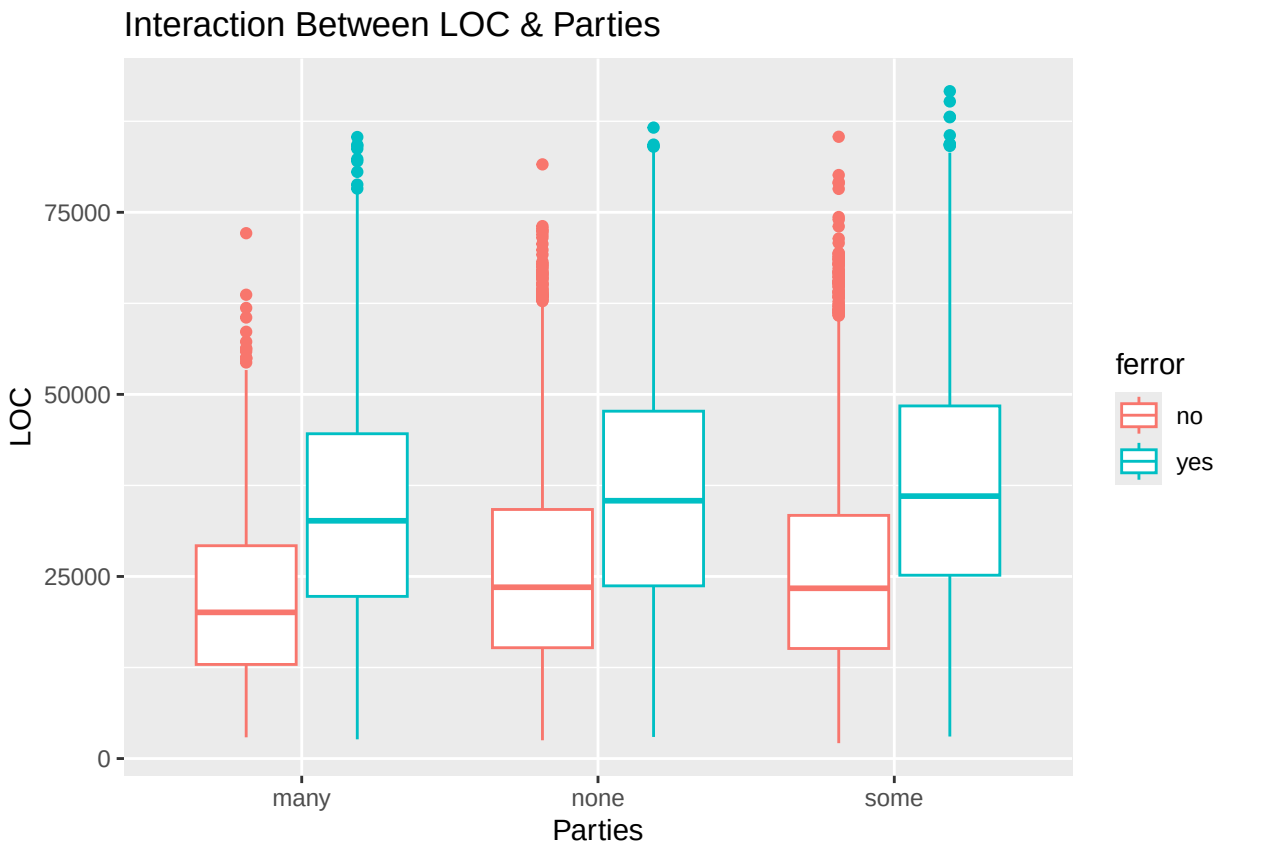


- According to the plot, we can see that the relationship between experience level, lines of code, and the occurrence of fatal errors varies significantly across different levels of experience. For the minimal experience group, the median LOC is lower compared to the moderate and extensive experience groups. Additionally, the proportion of fatal errors is higher for the minimal experience group. The extensive experience group shows higher variability in LOC, with a wider interquartile range, and a lower proportion of fatal errors compared to the minimal group.
- The amount of code written and the fatal errors occurence may be influenced by the experience level of the team leader. Team leader with extensive experience may be involved in more complex or larger projects. Given in higher LOC and greater variability, but they are also more effective in avoiding fatal errors. In contrast, those with minimal experience may work on simpler tasks, producing a lower LOC but facing a higher likelihood of fatal errors due to inexperience.
- In summary, more experienced team members might be trusted with tasks that require writing more lines of code, which could explain the higher LOC observed in the moderate and extensive groups. Their experience may also contribute to fewer fatal errors. Conversely, less experienced members may

produce more consistent but smaller outputs, leading to lower median LOC and a higher incidence of errors due to limited experience and familiarity with complex scenarios.

Second, we plot the interaction between loc and parties(loc:parties)

```
ggplot(data_training, aes(x = parties, y = loc, col = ferror)) +
  geom_boxplot() +
  labs(title = "Interaction Between LOC & Parties",
       x = "Parties",
       y = "LOC",
       fill = "FError")
```

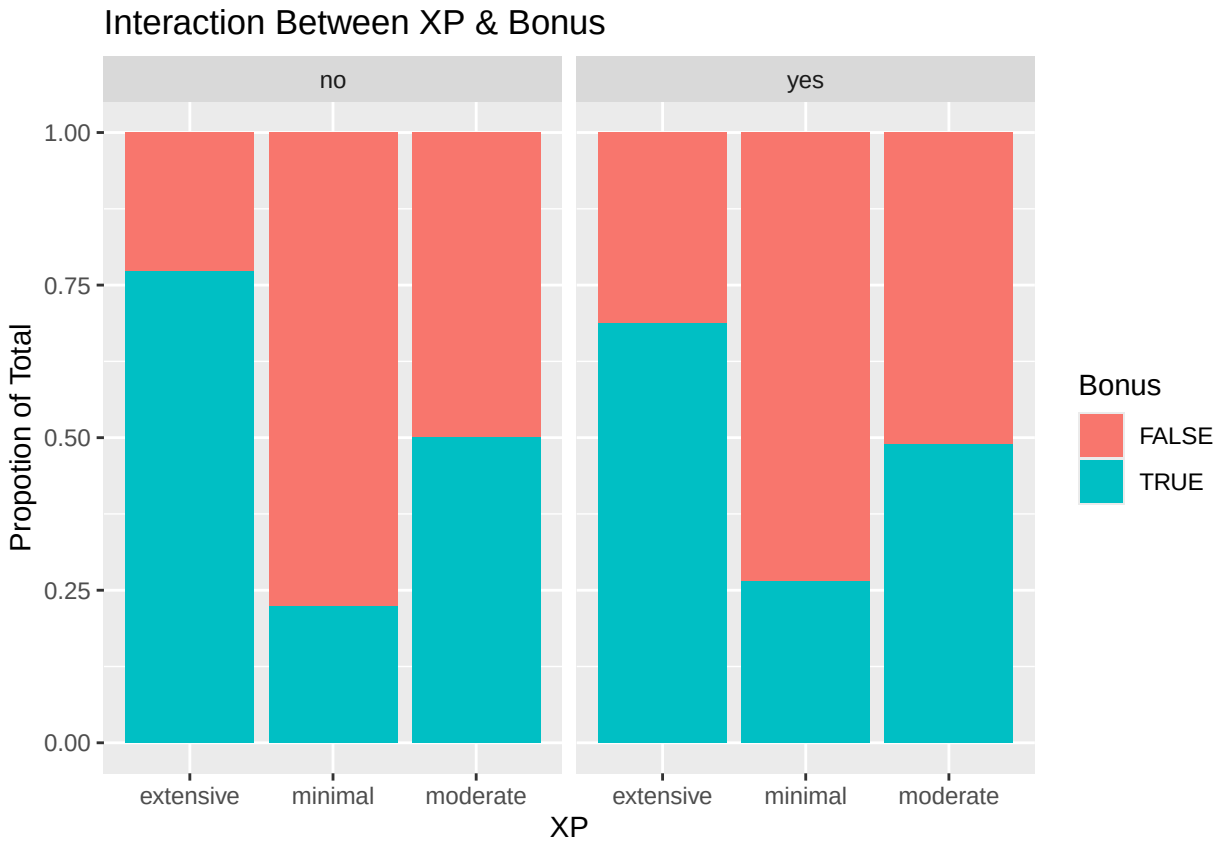


- The boxplot shows the relationship between the number of office parties and the lines of code (LOC), categorized by whether a fatal error occurred. The boxes represent the interquartile range, with the middle line indicating the median LOC. The colored boxes show the distinction between projects with and without fatal errors.
- A potential hypothesis is the occurrence of fatal errors may be influenced by the number of office parties and the volume of work . Teams that had many office parties tend to have a lower median LOC, but the possibility of fatal error occurs obviously lower than two other groups. This might shows that high workload combined with increased social activities could lead to higher stress and no enough time to create more code of lines. However, due to attending the office parties they obviously lower the rate of fatal error occurred. Maybe the parties helped to create a good team work. In contrast, teams with none or some office parties show higher fatal error occurance rate. Both groups median LOC are higher, possibly indicating better focus or fewer distractions during work. But lower the team integration to prevent fatal error occurred.
- In conclusion, office parties could have a dual impact on productivity. On one hand, they may enhance

team morale, leading to lower likelihood of error occurs. On the other hand, frequent social gatherings might also distract the team, resulting of low output (more lines of code). This dual effect can be observed in the variation of LOC and the presence of fatal errors across different levels of office parties.

Third, we plot the interaction between xp and bonus(xp:bonus)

```
# Here we plot bar chart with facet_wrap to show the propotion of receiving bonus
ggplot(data_training, aes(x = xp, fill = bonus)) +
  geom_bar(position = "fill") +
  labs(
    title = "Interaction Between XP & Bonus",
    x = "XP",
    y = "Propotion of Total",
    fill = "Bonus") +
  facet_wrap(~ferror)
```



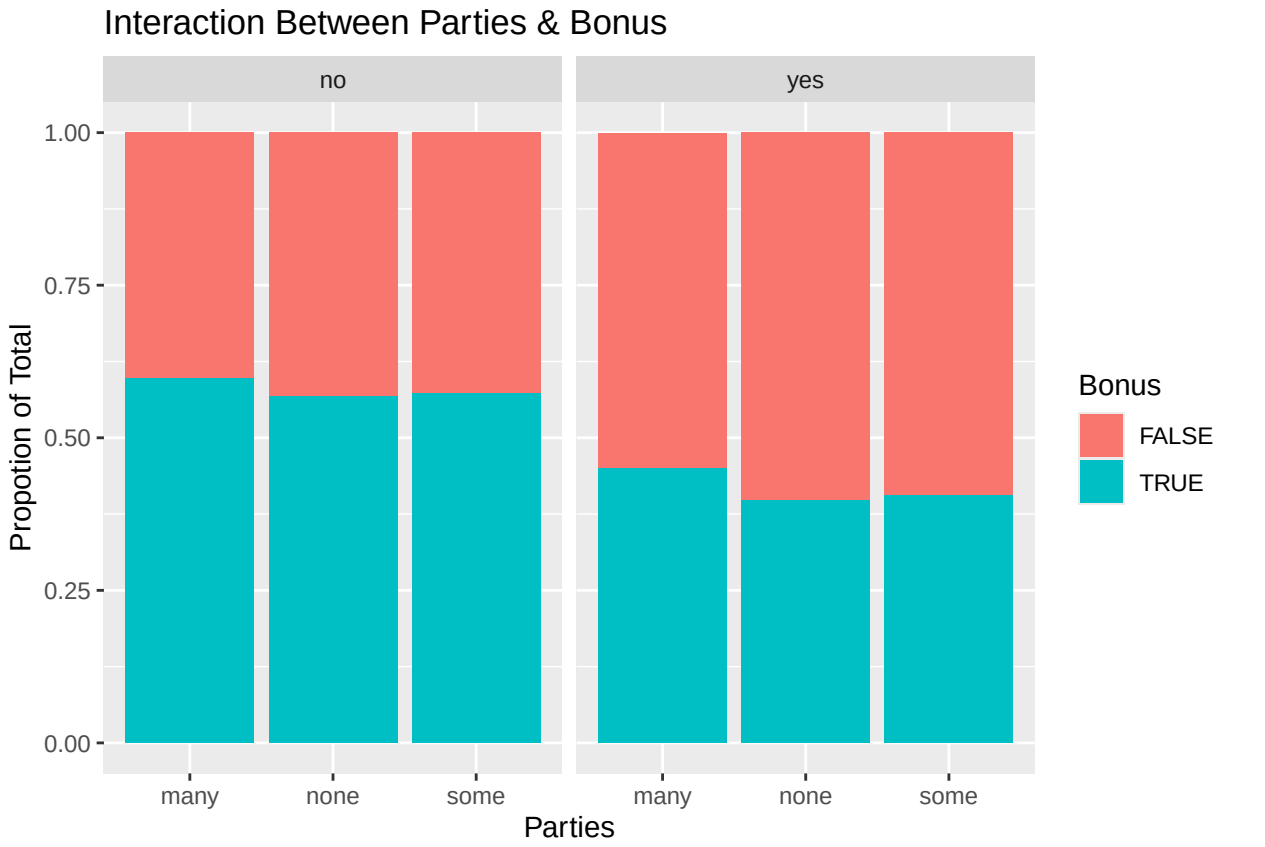
- According to the plot, we see that the relationship between experience level, bonus received, and the occurrence of fatal errors across different levels of experience. For the minimal experience team leader group, the proportion of individuals who received a bonus is lower compared to the other groups, regardless of whether a fatal error occurred. On the other hand, for the extensive experience team leader group, the percentage of individuals receiving a bonus is significantly higher, especially among those who did not experience a fatal error.
- Potential Hypothesis: More experienced team leaders are more likely to receive bonuses due to their efforts, greater productivity, or higher skill level, particularly when they avoid fatal errors. Their familiarity with expectations and ability to exceed them makes them more deserving of bonuses. By a contrast, team leaders with minimal experience may be less likely to receive bonuses, especially if they are associated with a higher probability of fatal errors. This could be because they are still learning

and might not be as effective as their experienced colleagues in avoiding mistakes.

- There is also a straightforward explanation for this trend that bonuses may be used as a motivational tool for experienced workers to maintain high performance, which is crucial for achieving project goals. For less experienced team leaders, the focus may be on training and development rather than immediate rewards, which could explain the lower distribution of bonuses, particularly if they are also associated with a higher rate of fatal errors. Inexperienced team leaders may need more time and guidance to improve their performance and error rates before becoming eligible for bonuses.

Forth, we plot the interaction between parties and bonus (parties:bonus)

```
ggplot(data_training, aes(x = parties, fill = bonus)) +
  geom_bar(position = "fill") +
  labs(
    title = "Interaction Between Parties & Bonus",
    x = "Parties",
    y = "Propotion of Total",
    fill = "Bonus") +
  facet_wrap(~ferror)
```



- In the plot, we observe the interaction between office parties, whether a bonus was received, and the occurrence of fatal errors. The plot shows how the distribution of bonuses changes depending on the number of office parties attended, while also differentiating between a fatal error occurred or did not occur. For each level of parties, we can see the proportional difference in the frequency of receiving a bonus across fatal error conditions.
- A potential hypothesis is that the occurrence of fatal errors might effect how bonuses are distributed, depending on participation in office parties. For instance, workers who attend many office parties and

experience fewer fatal errors may be more likely to receive bonuses due to improved performance, as their active social engagement could positively impact team cohesion and productivity. By a contrast, those who attend fewer or no office parties and experience fatal errors may be less likely to receive bonuses, potentially due to lower perceived contribution to team dynamics.

- For the interaction between office parties, bonuses, and fatal errors. Active participation in office events could contribute to better communication and morale within the team, leading to a lower likelihood of fatal errors. This might be different for people who increase the chances of receiving bonuses for those individuals. On the other hand, low participation in office activities might correlate with a higher incidence of errors, possibly due to weaker team integration, which could lead to fewer bonuses being awarded to such individuals.

Q10.

Logistic Regression Model for Log-Odds

- The logistic regression model estimates the log-odds of the probability of a fatal error:

For our model, the estimated function of predictors can be written as:

$$\begin{aligned}\hat{b}_i = \hat{\beta}_0 &+ \hat{\beta}_1 \cdot loc + \hat{\beta}_2 \cdot xp_{\text{minimal}} + \hat{\beta}_3 \cdot xp_{\text{moderate}} \\ &+ \hat{\beta}_4 \cdot parties_{\text{none}} + \hat{\beta}_5 \cdot parties_{\text{some}} \\ &+ \hat{\beta}_6 \cdot bonus_{\text{TRUE}} + \hat{\beta}_7 \cdot (loc \cdot xp_{\text{minimal}}) \\ &+ \hat{\beta}_8 \cdot (loc \cdot xp_{\text{moderate}}) + \hat{\beta}_9 \cdot (loc \cdot parties_{\text{none}}) \\ &+ \hat{\beta}_{10} \cdot (loc \cdot parties_{\text{some}}) \\ &+ \hat{\beta}_{11} \cdot (xp_{\text{minimal}} \cdot bonus_{\text{TRUE}}) \\ &+ \hat{\beta}_{12} \cdot (xp_{\text{moderate}} \cdot bonus_{\text{TRUE}}) \\ &+ \hat{\beta}_{13} \cdot (parties_{\text{none}} \cdot bonus_{\text{TRUE}}) \\ &+ \hat{\beta}_{14} \cdot (parties_{\text{some}} \cdot bonus_{\text{TRUE}})\end{aligned}$$

Definitions of Variables:

- $\hat{\beta}_0$: The estimated intercept.
- $\hat{\beta}_1$: The estimated coefficient for `loc`.
- $\hat{\beta}_2$: The estimated coefficient for `xp_{minimal}`.
- $\hat{\beta}_3$: The estimated coefficient for `xp_{moderate}`.
- $\hat{\beta}_4$: The estimated coefficient for `parties_{none}`.
- $\hat{\beta}_5$: The estimated coefficient for `parties_{some}`.
- $\hat{\beta}_6$: The estimated coefficient for `bonus_{TRUE}`.
- $\hat{\beta}_7$: The estimated coefficient for `loc:xp_{minimal}`.
- $\hat{\beta}_8$: The estimated coefficient for `loc:xp_{moderate}`.

- $\hat{\beta}_9$: The estimated coefficient for `loc:parties_{none}`.
- $\hat{\beta}_{10}$: The estimated coefficient for `loc:parties_{some}`.
- $\hat{\beta}_{11}$: The estimated coefficient for `xp_{minimal}:bonus_{TRUE}`.
- $\hat{\beta}_{12}$: The estimated coefficient for `xp_{moderate}:bonus_{TRUE}`.
- $\hat{\beta}_{13}$: The estimated coefficient for `parties_{none}:bonus_{TRUE}`.
- $\hat{\beta}_{14}$: The estimated coefficient for `parties_{some}:bonus_{TRUE}`.

Q11.

Q11.(a). How many lines described for final model

- The final logistic regression model includes multiple categorical and continuous predictors along with interaction terms. To determine the number of lines described by the model, we count the combinations of the categorical levels and continuous predictors:

Predictors:

- **`xp`**: Categorical variable with 3 levels ('minimal', 'moderate', 'extensive').
- **`parties`**: Categorical variable with 3 levels ('none', 'some', 'many').
- **`bonus`**: Logical variable with 2 levels ('TRUE', 'FALSE').
- **`loc`**: A continuous predictor.

$$Total\ lines: 3 * 3 * 2 = 18\ lines$$

- Since loc is continuous, each group corresponds to a line in the geometric space.
- For the final model, there are 18 unique lines based on the categorical combinations of xp, parties, and bonus.

Q11.(b). Are the lines all parallel

- No, not all the line are parallel due to the interaction terms in the model effect the slopes of the related lines. Interaction terms modify how two variables relate to each other, it means that the slope of one variable changes based on the value of the other variable. Thus the slope will move base on two variables. This causes to differing slopes for the lines and making them not parallel.
- About the interaction term loc:xp. Xp has three levels: xpminimal, xpmoderate, and xpextensive. When loc interacts with xp, the slope of `loc` relys on the different level of xp. These differences in slopes demonstrate that the lines are not parallel.
- In summary, interaction terms like loc:xp and xp:bonus cause different slopes in different levels of the interacting variables, which explains why the lines in the model are not parallel.

Q12.

```
# Output the final model by summary
summary(lr_model5$fit)
```

```
##
## Call:
```

```
## stats::glm(formula = ferror ~ loc + xp + parties + bonus + loc:xp +
##     loc:parties + xp:bonus + parties:bonus, family = stats::binomial,
##     data = data)
##
## Coefficients:
##                          Estimate Std. Error z value Pr(>|z|)
## (Intercept)           -2.409e+00  1.145e-01 -21.049  < 2e-16 ***
## loc                    8.438e-05  3.343e-06  25.240  < 2e-16 ***
## xpminimal              2.157e+00  1.063e-01  20.299  < 2e-16 ***
## xpmoderate             1.195e+00  1.087e-01  10.994  < 2e-16 ***
## partiesnone           -6.713e-01  1.076e-01  -6.240 4.38e-10 ***
## partiessome           -1.093e+00  1.011e-01 -10.808  < 2e-16 ***
## bonusTRUE             -4.435e-01  8.812e-02  -5.033 4.82e-07 ***
## loc:xpminimal         -9.118e-06  2.859e-06  -3.190 0.001424 **
## loc:xpmoderate        -4.835e-06  2.761e-06  -1.751 0.079961 .
## loc:partiesnone       -2.340e-05  3.476e-06  -6.732 1.67e-11 ***
## loc:partiessome       -1.166e-05  3.321e-06  -3.511 0.000447 ***
## xpminimal:bonusTRUE    8.596e-01  8.973e-02   9.580  < 2e-16 ***
## xpmoderate:bonusTRUE   5.428e-01  8.228e-02   6.597 4.20e-11 ***
## partiesnone:bonusTRUE -1.139e-01  9.104e-02  -1.252 0.210715
## partiessome:bonusTRUE -2.148e-01  8.506e-02  -2.526 0.011553 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 34570  on 24999  degrees of freedom
## Residual deviance: 25758  on 24985  degrees of freedom
## AIC: 25788
##
## Number of Fisher Scoring iterations: 5
```

- From the summary(), we substitute the estimated value into the equation $

$$
\begin{aligned}
\hat{b}_i = & -2.409 + 8.438 \times 10^{-5} \cdot loc_i + 2.157 \cdot xpminimal_i \\
& + 1.195 \cdot xpmoderate_i - 0.6713 \cdot partiesnone_i \\
& - 1.093 \cdot partiessome_i - 0.4435 \cdot bonusTRUE_i \\
& - 9.118 \times 10^{-6} \cdot (loc \cdot xpminimal_i) \\
& - 4.835 \times 10^{-6} \cdot (loc \cdot xpmoderate_i) \\
& - 2.340 \times 10^{-5} \cdot (loc \cdot partiesnone_i) \\
& - 1.166 \times 10^{-5} \cdot (loc \cdot partiessome_i) \\
& + 0.8596 \cdot (xpminimal \cdot bonusTRUE_i) \\
& + 0.5428 \cdot (xpmoderate \cdot bonusTRUE_i) \\
& - 0.1139 \cdot (partiesnone \cdot bonusTRUE_i) \\
& - 0.2148 \cdot (partiessome \cdot bonusTRUE_i)
\end{aligned}
$$

Q13. Scenario

Q13.(a).

- **Foosball table**: Foosball table is not related because it was removed from the final model.

- **Experience level**: Highly experienced team leader is 0 as it is the reference level.
- **Work functions**: The team does not attend any work functions.
- **Bonuses**: The team received bonuses last year.
- The estimated log-odds for condition as follow:

$$\begin{aligned}\hat{b}_i &= \hat{\beta}_0 + \hat{\beta}_1 \cdot loc_i + \hat{\beta}_4 \\ &\quad + \hat{\beta}_6 + \hat{\beta}_9 + \hat{\beta}_{13} \\ &= -2.409 + 8.438 \times 10^{-5} \cdot loc - 0.6713 - 0.4435 - 0.0000234 \\ &\quad - 0.1139 \\ &= -3.6377234 + 0.00008438 \cdot loc\end{aligned}$$

Q13.(b).

- **Foosball table**: Foosball table is not related because it was removed from the final model.
- **Experience level**: Moderate experienced team leader.
- **Work functions**: The team attend some work functions.
- **Bonuses**: The team did not received bonuses.
- The estimated log-odds for condition as follow:

$$\begin{aligned}\hat{b}_i &= \hat{\beta}_0 + \hat{\beta}_1 \cdot loc_i + \hat{\beta}_3 \\ &\quad + \hat{\beta}_5 + \hat{\beta}_8 + \hat{\beta}_{10} \\ &= -2.409 + 8.438 \times 10^{-5} \cdot loc + 1.195 - 1.093 - 0.000004835 \\ &\quad - 0.00001166 \\ &= -2.307016495 + 0.00008438 \cdot loc\end{aligned}$$

Q14. Using test data to predict

```
testing_predictions <- data_testing %>%
  select(ferror) %>%
  bind_cols(
    predict(
      object = lr_model5,
      new_data = data_testing,
      type = "class"
    ) %>%
      rename(predicted_class = .pred_class)
  ) %>%
  bind_cols(
    predict(
      object = lr_model5,
      new_data = data_testing,
      type = "prob"
    ) %>%
      rename(prediction_prob = .pred_yes)
```

```
  )
# Display first 10 outputs
head(testing_predictions, 10)
```

```
## # A tibble: 10 x 4
##    ferror predicted_class .pred_no prediction_prob
##    <fct>  <fct>              <dbl>           <dbl>
##  1 no     no                 0.809          0.191
##  2 yes    no                 0.537          0.463
##  3 no     no                 0.535          0.465
##  4 yes    no                 0.589          0.411
##  5 yes    no                 0.589          0.411
##  6 yes    no                 0.560          0.440
##  7 yes    yes                0.356          0.644
##  8 no     no                 0.928          0.0718
##  9 yes    yes                0.365          0.635
## 10 no     no                 0.866          0.134
```

```
# Display the dimensions for prediction
dim(testing_predictions)
```

```
## [1] 11000     4
```

Q15. Evaluate the model

Q15.(a). Confusion matrix

```
# Display confusion matrix
testing_predictions %>%
  conf_mat(truth = ferror, estimate = predicted_class)
```

```
##           Truth
## Prediction   no  yes
##        no  4563 1513
##        yes 1408 3516
```

```
# Using accuracy() to get the accuracy
testing_predictions %>%
  accuracy(truth = ferror, estimate = predicted_class)
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.734
```

Q15.(b).Sensitivity and specificity

```
# Here we calculate sensitivity manually, we got true positive 4563 and false positive 1408 from confus
sens <- 4563 / (4563 + 1408)
sens
```

```
## [1] 0.7641936
```

```
# Here we calculate specificity manually, we got true negative 3516 and false negative 1513 from confus
spec <- 3516 / (1513 + 3516)
```

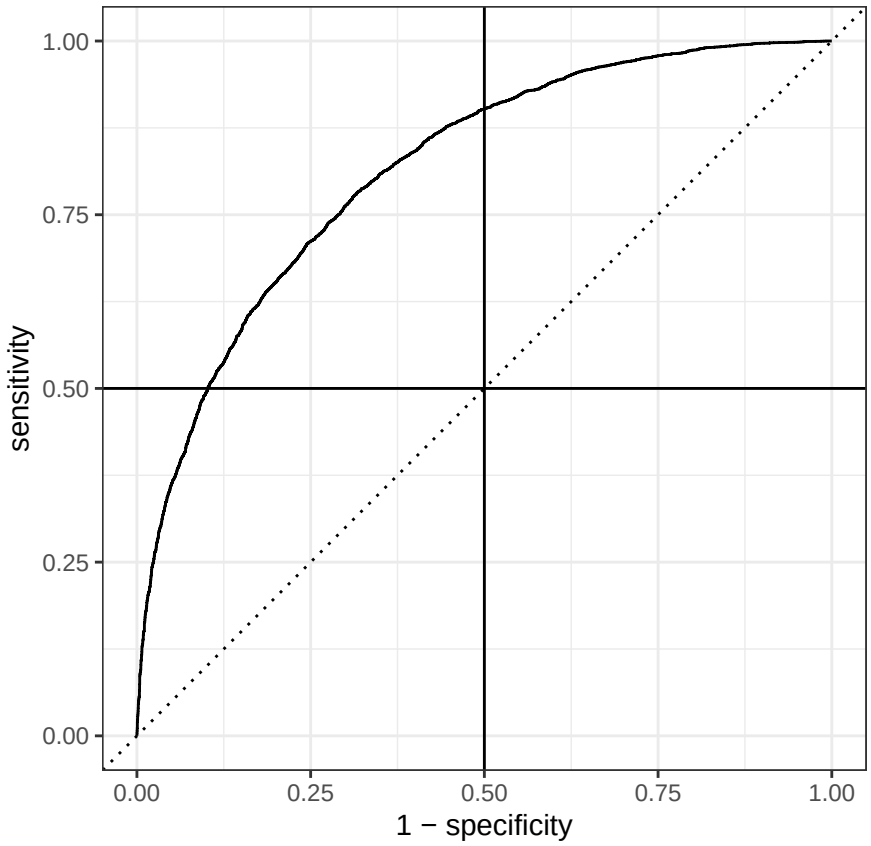
```
spec
```

```
## [1] 0.699145
```

- Sensitivity is 0.764, specificity is 0.699

Q15.(c).ROC curve

```
testing_predictions %>%
  roc_curve(truth = ferror, .pred_no) %>%
  autoplot() +
  geom_vline(xintercept = 0.5) +
  geom_hline(yintercept = 0.5)
```



Q15.(d).The AUC of this ROC curve

```
testing_predictions %>%
  roc_auc(truth = ferror, .pred_no)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc binary         0.815
```

- The AUC of this ROC curve is 0.815

Q16. Predict for Bob's team

In the background for Bob's team is below:

- **Foosball table**: Foosball table is mentioned but it is not related because it was removed from the final model.
- **Lines of code**: 100,000 lines of code
- **Experience level**: Minimal experienced for team leader because Bob is junior team leader.
- **Parties**: Bob has promised to take his team with him to all of the events. And many parties is reference level, thus the value is 0.
- **Bonuses**: The team members all received bonuses last year.
- According to the situation:

```
# Here we use our model to predict Bob's team
bob_team <- tibble(
  loc = 100000,
  xp = "minimal",
  foosball = TRUE,
  parties = "many",
  bonus = TRUE
)

bob_team_prediction <- bob_team %>%
  bind_cols(
    predict(
      object = lr_model5,
      new_data = bob_team,
      type = "class"
    ) %>%
      rename(predicted_class = .pred_class)
  ) %>%
  bind_cols(
    predict(
      object = lr_model5,
      new_data = bob_team,
      type = "prob"
    ) %>%
      rename(prediction_prob = .pred_yes)
  )
bob_team_prediction
```

```
## # A tibble: 1 x 8
##      loc xp      foosball parties bonus predicted_class .pred_no prediction_prob
##    <dbl> <chr>   <lgl>    <chr>   <lgl> <fct>              <dbl>           <dbl>
## 1 100000 minimal TRUE     many    TRUE  yes             0.000457            1.00
```

- Yes, based on our model, it is very likely that Bob's team will encounter a fatal error while debugging the final software program.

*According to our logistic regression model, which takes into consideration various team factors such as lines of code (LOC), team experience (XP), office activities, and bonus incentives, our prediction shows a high probability of fatal errors in Bob's team scenario.

- Our model predicts a near 100% probability of Bob's team missing a fatal error in this project. The

output suggests a very high likelihood of a critical failure, which poses a significant risk to project stability.