# HW3_UPDATED_starter_template_R_SOLUTIONS_V6e

June 27, 2025

[ ]:
```
---
title: "HW3_starter_template"
output: pdf_document
---
```

### 0.1 Part 1

### 0.2 Data Description

The data contains characteristics of employees at a company.

**Age**: (Qualitative) Age group 2-5 (2 corresponds to 20s, 3 to 30s, etc.)

**Gender**: (Qualitative) 1 for male, 0 for female

**Years**: (Quantitative) Number of years with the company

**Level**: (Qualitative) Seniority level of employee

**Pension**: (Qualitative) 1 for has pension, 0 for no pension

**Stayed**: Number of employees who stayed with the company

**Total**: Total Number of employees

We will use the variables **Stayed** and **Total** to create the response variable, **Retention**.

### 0.3 Data Preparation

Do not change this code cell:

[1]:
```
set.seed(100) # Do not change!
# Import the data
retention = read.csv("retention.csv", header=TRUE)

# Create response variable
retention$Retention = retention$Stayed/retention$Total

# Factorize categorical variables
retention$Age<-as.factor(retention$Age)
retention$Gender<-as.factor(retention$Gender)
retention$Level<-as.factor(retention$Level)
retention$Pension<-as.factor(retention$Pension)
```

```
head(retention)
```

A data.frame: 6 × 8

| | Age <fct> | Gender <fct> | Years <int> | Level <fct> | Pension <fct> | Stayed <int> | Total <int> | Retention <dbl> |
|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 1 | 3 | 1 | 0 | 6 | 12 | 0.5000000 |
| 2 | 2 | 1 | 4 | 1 | 0 | 6 | 11 | 0.5454545 |
| 3 | 2 | 1 | 4 | 1 | 1 | 3 | 14 | 0.2142857 |
| 4 | 2 | 0 | 7 | 1 | 0 | 4 | 11 | 0.3636364 |
| 5 | 2 | 1 | 7 | 1 | 0 | 3 | 15 | 0.2000000 |
| 6 | 2 | 0 | 4 | 2 | 0 | 5 | 13 | 0.3846154 |

### 0.3.1 Question 1 - Model Fit & Interpretation (6 points)

(a) 2 pts - Fit a logistic regression model (model1) using **Pension** as the predicting variable and **Retention** as the response variable. Display model summary and explain each parameter you used in the glm function in your words.

[2]:
```
set.seed(100) # Do not change!
model1=glm(Retention~Pension,data=retention, family="binomial", weights=Total)
summary(model1)
```

```
Call:
glm(formula = Retention ~ Pension, family = "binomial", data = retention,
    weights = Total)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-5.5220  -1.6169  -0.2689   2.0749   4.8391

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  0.37285    0.05121   7.281 3.31e-13 ***
Pension1    -0.78974    0.07873 -10.031  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 915.00  on 157  degrees of freedom
Residual deviance: 812.34  on 156  degrees of freedom
AIC: 1261.7

Number of Fisher Scoring iterations: 4
```

Q1a Answer:
formula - the formula for the logistic regression model (left side is response, right side is predictor)

family - the type of model (Binomial for logistic regression)

data - the data for model fitting

weights - number of trials (or repetitions) for each aggregated row

(b) 1 pt - What is the estimated equation for the probability of an employee staying with the company (retention)?

Q1b Answer:
Let p be the probability of an employee staying with the company (Retention) given the predicting variable Pension[T.1], then the equation for p is:

exp(0.37285-0.78974X_Pension[T.1]) / (1 + exp(0.37285-0.78974X_Pension[T.1]))

(c) 1 pt - Interpret the estimated coefficient for **Pension** with respect to the log-odds of retention.

Q1c Answer:
The log-odds of retention decreases by 0.78974 for employees with pension (1) versus those without pension (0).

(d) 1 pt - Interpret the estimated coefficient for **Pension** with respect to the odds of retention.

Q1d Answer:
The odds of retention for employees with pension (1) are 1-exp(-0.78974) = 0.546 lower than for those without pension (0).

OR

Compared to those without pension (0), the odds of retention for employees with pension (1) changes by a factor of exp(-0.78974) = 0.454.

(e) 1 pt - Explain why we can or cannot perform a goodness of fit test for model1. No need to perform the GOF test.

Q1e Answer:
We can perform the GOF test for model1 because the dataset has repetitions (binomial data).

### 0.3.2 Question 2 - Full Model & Assessment (6 points)

(a) 1 pt - Fit a logistic regression model (model2) using **Age**, **Gender**, **Years**, **Level**, and **Pension** as the predicting variables and **Retention** as the response variable. Display model summary.

[3]:
```
set.seed(100) # Do not change!
model2=glm(Retention~Age+Gender+Years+Level+Pension,data=retention,␣
  ↪family="binomial", weights=Total)
summary(model2)
```

```
Call:
glm(formula = Retention ~ Age + Gender + Years + Level + Pension,
    family = "binomial", data = retention, weights = Total)

Deviance Residuals:
```

```
     Min        1Q    Median        3Q       Max
-2.92330  -0.68789  -0.07553   0.71082   2.86697

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept)  8.212e-02  2.580e-01   0.318    0.750
Age3         2.562e-01  2.429e-01   1.055    0.292
Age4         1.526e+00  2.459e-01   6.205 5.48e-10 ***
Age5         2.768e+00  3.179e-01   8.709  < 2e-16 ***
Gender1     -5.340e-01  8.904e-02  -5.997 2.01e-09 ***
Years        1.106e-05  1.576e-02   0.001    0.999
Level2      -1.207e+00  1.019e-01 -11.843  < 2e-16 ***
Pension1    -7.521e-01  9.047e-02  -8.313  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 915.00  on 157  degrees of freedom
Residual deviance: 150.85  on 150  degrees of freedom
AIC: 612.19

Number of Fisher Scoring iterations: 4
```

(b) 1 pt - Perform the test for overall significance for model2 and interpret the result using alpha = 0.05.

[4]:
```
set.seed(100) # Do not change!
1-pchisq(model2$null.deviance-model2$deviance, model2$df.null-model2$df.resid)
```

0

Q2b Answer:
The p-value for the chi-squared test is zero, so the overall regression is significant and model2 has explanatory power.

(c) 1 pt - Perform the goodness of fit test using deviance and Pearson residuals and interpret the results.

[5]:
```
# Deviance residuals test
set.seed(100) # Do not change!
cat("p-value of Deviance residuals test is:", 1-pchisq(model2$deviance,␣
 ↪model2$df.residual), end="\n")
```

```
p-value of Deviance residuals test is: 0.4652562
```

[6]:
```
# Pearson residuals test
set.seed(100) # Do not change!
pResid <- resid(model2, type = "pearson")
```

```
cat("p-value of Pearson residuals test is:", 1-pchisq(sum(pResid^2), model2$df.
  ↪residual))
```

```
p-value of Pearson residuals test is: 0.6672298
```

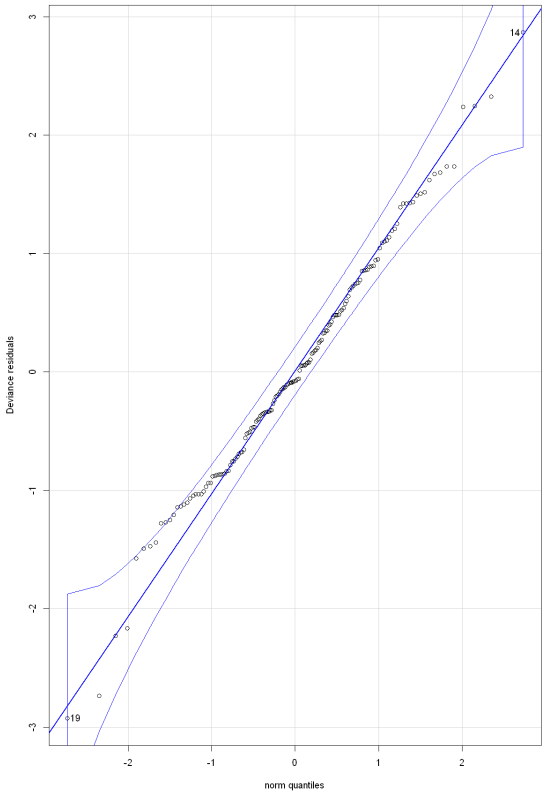
Q2c Answer:
The null hypothesis is the model has good fit. Since the p-values for both tests are large, we do not reject the null hypothesis and conclude that model2 has good fit.

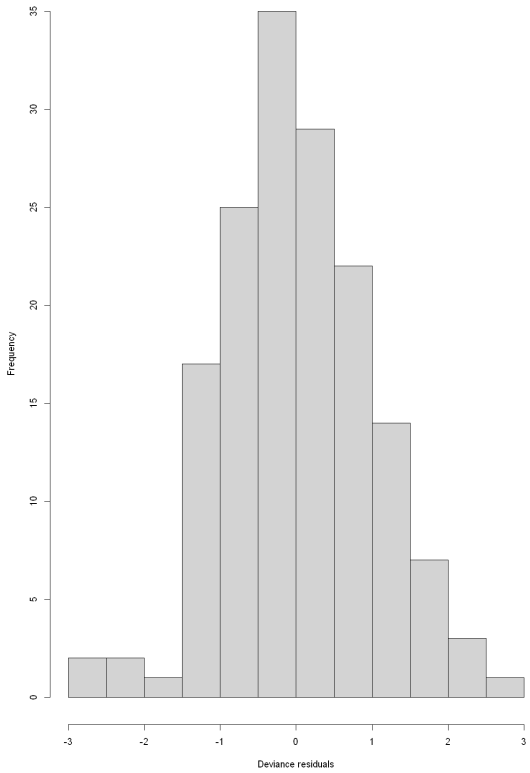
(d) 2 pts - Use both a QQ Plot and a histogram to evaluate if the deviance residuals are normally distributed. Interpret both plots.

[7]:
```
library(repr)
# Set custom figure size
options(repr.plot.width = 20, repr.plot.height = 15)

set.seed(100) # Do not change!
library(car)
res = resid(model2,type="deviance")

par(mfrow=c(1,2))
qqPlot(res, ylab="Deviance residuals")
hist(res, 10, xlab="Deviance residuals", main="")
```

```
Loading required package: carData
```

1. 19 2. 14




Q2d Answer:
The QQ Plot shows that most of the residuals fall along the reference line and within the 95% confidence band with very few points deviating on the left side. The histogram confirms this observation with a mostly normal distribution and very few stray points on the lower end. Using the combination of both plots, it appears the deviance residuals follow a normal distribution.

(e) 1 pt- Based on the GOF tests (2c) and the plots (2d), is model2 a good fit for the data?

Q2e Answer:
Yes, model2 is a good fit for the data.

### 0.3.3 Question 3 - Prediction (1 point)

Using the following data point:

- Age: 5
- Gender: 0 (Female)
- Years: 3
- Level: 1
- Pension: 1

(a) 1 pt - Predict this employee's probability of staying with the company (retention) using model2.

[8]:
```r
set.seed(100) # Do not change!
new = data.frame(Age=5,
                 Gender=0,
                 Years=3,
                 Level=1,
                 Pension=1)

new$Age<-as.factor(new$Age)
new$Gender<-as.factor(new$Gender)
new$Level<-as.factor(new$Level)
new$Pension<-as.factor(new$Pension)

predict(model2, new, type="response")
```

**1:** 0.89074054720168

Q3a Answer:
A female employee in her 50s with 3 years of experience, a seniority level of 1, and pension has an estimated probability of 0.89 of staying with the company.

### 0.4 Data Description

The data contains the QC test results and characteristics of a medical device.

**Type**: (Qualitative) Device type, A or B

**Manufacturer**: (Qualitative) Manufacturer of device, X or Y

**Mass**: (Quantitative) Device mass (g)

**Length**: (Quantitative) Device length (mm)

**Thickness**: (Quantitative) Device thickness (mm)

**Result**: (Binary response) test result, 1 for pass and 0 for fail

### 0.5 Data Preparation

Do not change this code cell:

[9]:
```r
set.seed(100) # Do not change!

# Import the data
data = read.csv("medical.csv", header=TRUE)

# Factorize categorical variables
data$Type<-as.factor(data$Type)
data$Manufacturer<-as.factor(data$Manufacturer)

# Divide the dataset into training and testing datasets
testRows = sample(nrow(data), 0.2*nrow(data))
testData = data[testRows, ]
```

```
trainData = data[-testRows, ]

head(trainData)
```

A data.frame: 6 × 6

| | Type <fct> | Manufacturer <fct> | Mass <dbl> | Length <dbl> | Thickness <dbl> | Result <int> |
|---|---|---|---|---|---|---|
| 2 | Type B | Manufacturer X | 198.79 | 87.8 | 5.8 | 1 |
| 3 | Type A | Manufacturer X | 462.50 | 55.0 | 8.8 | 0 |
| 4 | Type A | Manufacturer X | 199.82 | 77.5 | 4.7 | 1 |
| 5 | Type A | Manufacturer X | 208.78 | 74.3 | 9.0 | 1 |
| 6 | Type B | Manufacturer X | 403.76 | 90.0 | 2.7 | 0 |
| 8 | Type A | Manufacturer X | 410.68 | 40.7 | 8.8 | 1 |

[10]:
```
write.csv(testData, "testData_R.csv", row.names=FALSE)
write.csv(trainData, "trainData_R.csv",row.names=FALSE)
```

### 0.5.1 Question 4 - Exploratory Analysis (2 points)

(a) 2 pts - Using trainData, create well-labeled stacked bar plots showing the proportion of devices that passed the QC test (green color) and those that failed (red color) for the two categorical predictors: **Type** and **Manufacturer**. Interpret the plots.

Note: response should be on the y-axis and predictor on the x-axis.

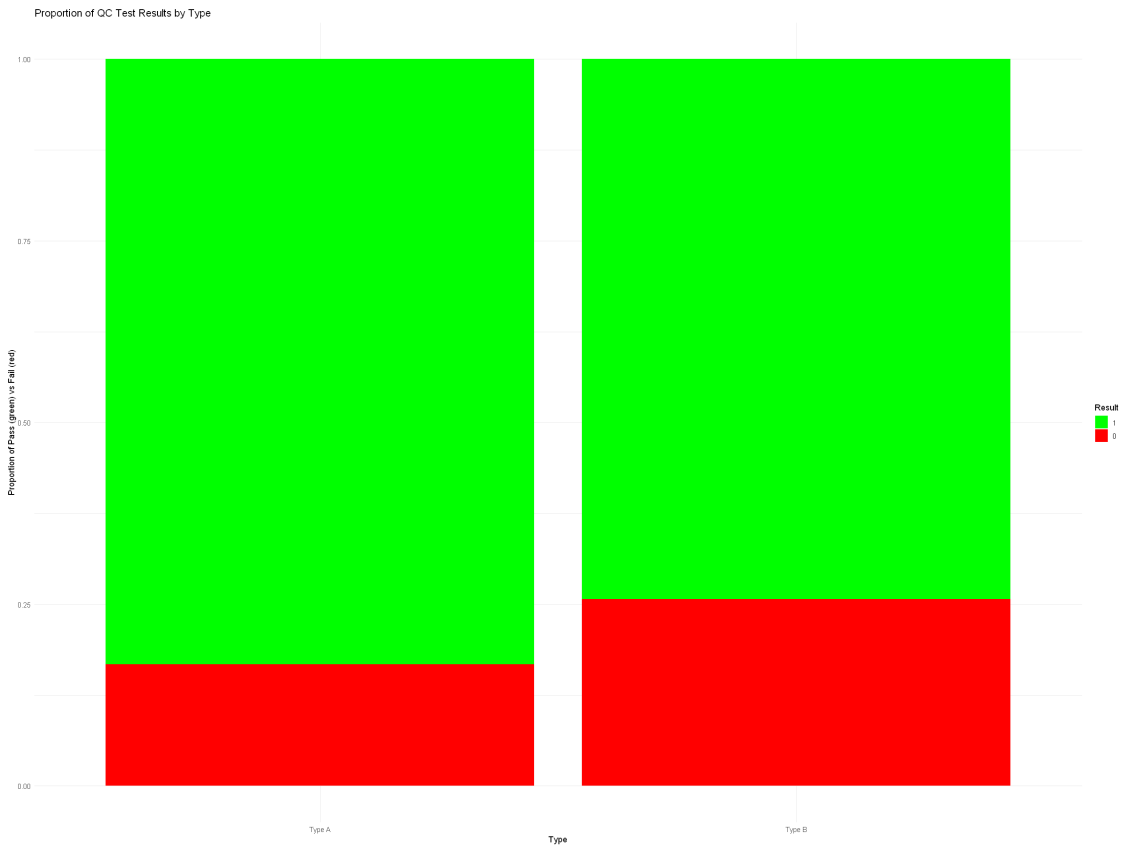
[11]:
```
set.seed(100) # Do not change!
library(tidyverse)
# Summarize the data by type and qc_test and calculate the proportion
type_prop <-  trainData %>%
  group_by(Type, Result) %>%
  summarise(count = n(), .groups = 'drop') %>%
  ungroup() %>%
  group_by(Type) %>%
  mutate(proportion = count / sum(count))

Manufacturer_prop <-  trainData %>%
  group_by(Manufacturer, Result) %>%
  summarise(count = n(), .groups = 'drop') %>%
  ungroup() %>%
  group_by(Manufacturer) %>%
  mutate(proportion = count / sum(count))

print(Manufacturer_prop)
```

```
-- Attaching packages
--------------------------------------------------------------------------------
tidyverse 1.3.2 --
v ggplot2 3.4.2     v purrr   1.0.2
v tibble  3.2.1     v dplyr   1.1.4
v tidyr   1.3.1     v stringr 1.5.0
```

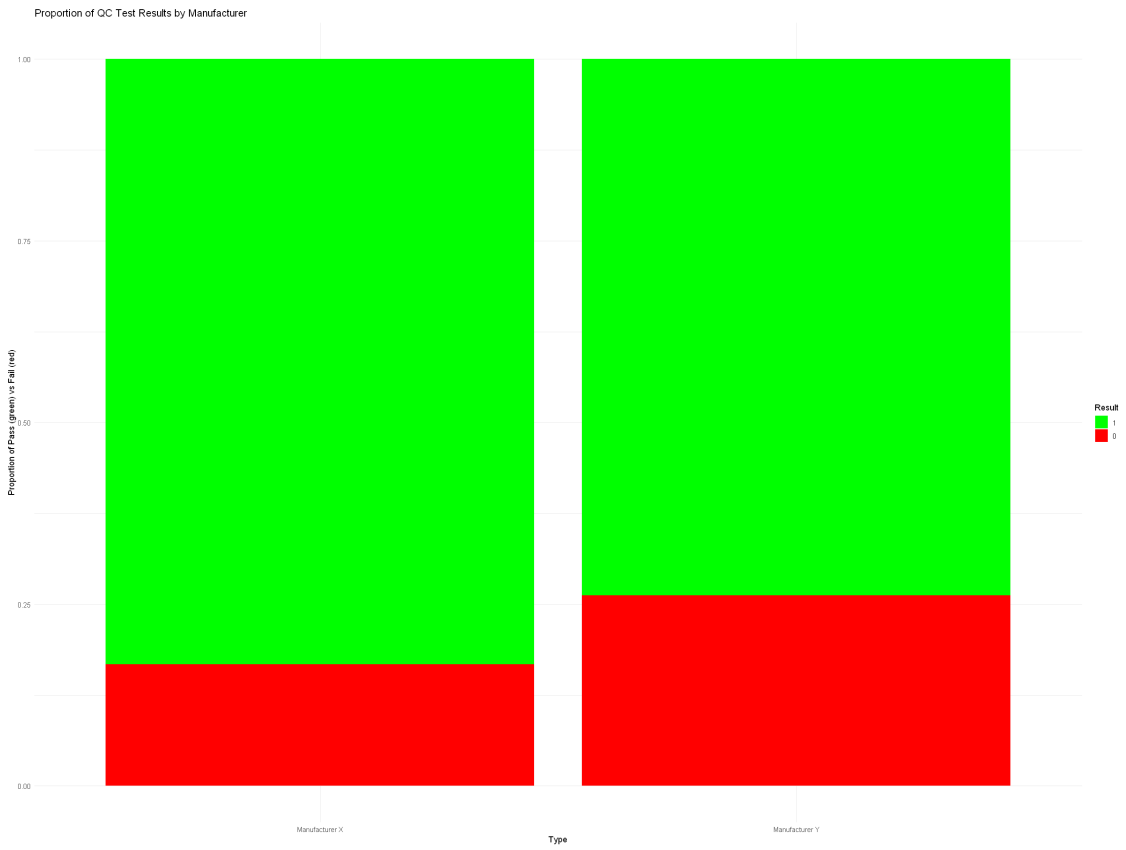
```
v readr   2.1.5     v forcats 1.0.0
-- Conflicts ---------------------------------------------------------
------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
x dplyr::recode() masks car::recode()
x purrr::some()   masks car::some()
```

```
# A tibble: 4 x 4
# Groups:   Manufacturer [2]
  Manufacturer   Result count proportion
  <fct>           <int>
<int>      <dbl>
1 Manufacturer X      0    70      0.167
2 Manufacturer X      1   349      0.833
3 Manufacturer Y      0   100      0.262
4 Manufacturer Y      1   281      0.738
```

[12]:
```
set.seed(100) # Do not change!
# Create the stacked bar plot showing proportions
# Type
ggplot(type_prop, aes(x = Type, y = proportion, fill = factor(Result, level =␣
 ↪c(1, 0)))) +
  geom_bar(stat = "identity") +
  scale_fill_manual(values = c("0" = "red", "1" = "green")) +  # Red for fail,␣
 ↪Green for pass
  labs(
    title = "Proportion of QC Test Results by Type",
    x = "Type",
    y = "Proportion of Pass (green) vs Fail (red)",
    fill = "Result"
  ) +
  theme_minimal()
```




[13]:
```r
set.seed(100) # Do not change!
# Manufacturer
ggplot(Manufacturer_prop, aes(x = Manufacturer, y = proportion, fill =␣
↪factor(Result, level = c(1, 0)))) +
 geom_bar(stat = "identity") +
 scale_fill_manual(values = c("0" = "red", "1" = "green")) +  # Red for fail,␣
↪Green for pass
 labs(
   title = "Proportion of QC Test Results by Manufacturer",
   x = "Type",
   y = "Proportion of Pass (green) vs Fail (red)",
   fill = "Result"
 ) +
 theme_minimal()
```




Q4a Answer:
Device Type B has a higher proportion of QC Test failures than device Type A. Devices from Manufacturer Y have higher proportion of QC Test failures than devices from Manufacturer X. Overall more devices pass QC tests regardless of Type or Manufacturer.

### 0.5.2 Question 5 - Model Fit & Evaluation (4 points)

(a) 1 pt - Using trainData and 5-fold Cross Validation, fit/train a Decision Tree model called model3 using **Mass**, **Length**, and **Thickness** as the predicting variables and **Result** as the response.

[14]:
```
set.seed(100) # Do not change!
library(caret)

col_names <- c("Mass", "Length", "Thickness", "Result")

# factorize Result for classification
trainData$Result <- as.factor(trainData$Result)

model3 <- caret::train(Result ~ .,
                       data = trainData[col_names],
```

```
                method = "rpart",
                trControl = trainControl(method = "cv", number = 5),
                metric = "Accuracy")
print(model3)
```

```
Loading required package: lattice


Attaching package: 'caret'


The following object is masked from 'package:purrr':

    lift



CART

800 samples
  3 predictor
  2 classes: '0', '1'

No pre-processing
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 640, 640, 640, 640, 640
Resampling results across tuning parameters:

  cp           Accuracy  Kappa
  0.005882353  0.78000    0.0169694398
  0.007843137  0.78375   -0.0006779661
  0.017647059  0.78750    0.0000000000

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was cp = 0.01764706.
```

(b) 1 pt - Display the average CV accuracy for model3.

[15]:
```
set.seed(100) # Do not change!
mean(model3$resample$Accuracy)
```

0.7875

(c) 1 pt - Using trainData and 5-fold Cross Validation, fit/train a logistic regression model called model4 using **Mass**, **Length**, and **Thickness** as the predicting variables and **Result** as the response.

[16]:
```
set.seed(100) # Do not change!

model4 <- caret::train(Result ~ .,
```

```
                data = trainData[col_names],
                method = "glm",
                trControl = trainControl(method = "cv", number = 5),
                metric = "Accuracy")
print(model4)
```

```
Generalized Linear Model

800 samples
  3 predictor
  2 classes: '0', '1'

No pre-processing
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 640, 640, 640, 640, 640
Resampling results:

  Accuracy  Kappa
  0.7875    0
```

(d) 1 pt - Display the average CV accuracy for model4. Using mean accuracy, which model (model3 or model4) is better?

[17]:
```
set.seed(100) # Do not change!
mean(model4$resample$Accuracy)
```

0.7875

Q5d Answer:
Both are equally accurate (mean accuracy of model3 and model 4 = 0.7875)

### 0.5.3 Question 6 - Prediction (3 points)

(a) 2 pts - Using testData, predict the QC test result (1 or 0) and display the first 5 rows of the classification using model3 and model4 and the actual Result. Use a threshold of 0.5, if necessary.

[18]:
```
set.seed(100) # Do not change!
# factorize Result for classification
testData$Result<-as.factor(testData$Result)

pred3 = predict(model3, testData[col_names])
pred4 = predict(model4, testData[col_names])

head(data.frame(pred3, pred4, testData$Result), 5)
```

A data.frame: 5 × 3

| | pred3 <fct> | pred4 <fct> | testData.Result <fct> |
|---|---|---|---|
| 1 | 1 | 1 | 0 |
| 2 | 1 | 1 | 1 |
| 3 | 1 | 1 | 1 |
| 4 | 1 | 1 | 1 |
| 5 | 1 | 1 | 0 |

(b) 1 pt - Based on the first 5 rows, which model is more accurate (model3 or model4)?

Q6b Answer:
Neither model3 nor model4 is more accurate; each model incorrectly classified 2 data points.

## 0.6 Data Description

The data contains the characteristics of a heating process and the number of defects found on the processed device component.

**Temperature**: (Quantitative) heating temperature (Celsius)

**Time**: (Quantitative) heating duration (min)

**Operator**: (Qualitative) operator who worked process (1 or 2)

**Defects**: number of defects

## 0.7 Data Preparation

Do not change this code cell:

[19]:
```
set.seed(100) # Do not change!
# Import the data
defects = read.csv("defects.csv", header=TRUE)

# Factorize categorical variables
defects$Operator<-as.factor(defects$Operator)

head(defects)
```

A data.frame: 6 × 4

| | Temperature <dbl> | Time <dbl> | Operator <fct> | Defects <int> |
|---|---|---|---|---|
| 1 | 104.97 | 24.66 | 2 | 3 |
| 2 | 98.62 | 21.59 | 2 | 1 |
| 3 | 106.48 | 24.14 | 2 | 1 |
| 4 | 115.23 | 22.00 | 1 | 1 |
| 5 | 97.66 | 28.27 | 1 | 3 |
| 6 | 97.66 | 29.53 | 2 | 4 |

### 0.7.1 Question 7 - Model Fit & Interpretation (3 points)

(a) 1 pt - Fit a poisson regression model (model5) using **Temperature**, **Time**, and **Operator** as the predicting variables and **Defects** as the response variable. Display model summary.

[20]:
```
set.seed(100) # Do not change!
model5 <- glm(Defects~Temperature+Time+Operator, data=defects, family="poisson")
summary(model5)
```

```
Call:
glm(formula = Defects ~ Temperature + Time + Operator, family = "poisson",
    data = defects)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.3066  -0.7596  -0.0630   0.5760   3.8291

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -0.799871   0.223649  -3.576 0.000348 ***
Temperature  0.004792   0.001619   2.960 0.003076 **
Time         0.038806   0.006222   6.237 4.46e-10 ***
Operator2   -0.011311   0.032028  -0.353 0.723966
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 2282.7  on 1999  degrees of freedom
Residual deviance: 2233.0  on 1996  degrees of freedom
AIC: 6729.2

Number of Fisher Scoring iterations: 5
```

(b) 1 pt - Interpret the estimated coefficient of **Time** with respect to the log expected Defect count.

Q7b Answer:
For every 1 min increase in heating Time, the log expected Defect count increases by 0.0388, provided all other predictors are held constant.

(c) 1 pt - Interpret the estimated coefficient of **Operator** with respect to the rate ratio for Defects.

Q7c Answer:
The rate ratio for Defects would be expected to decrease by a factor of exp(-0.0113)=0.9888 for components produced by Operator 2 compared to Operator 1, provided all other predictors are held constant.

### 0.7.2 Question 8 - Model Assessment (3 points)

(a) 1 pt - Perform the test for overall significance for model5 and interpret the result using alpha = 0.05.

[21]:
```
set.seed(100) # Do not change!
1-pchisq(model5$null.deviance-model5$deviance,
         model5$df.null-model5$df.resid)
```

9.22623089039121e-11

Q8a Answer:
The p-value for the chi-square test is close to 0, indicating that the overall regression is significant and model5 has explanatory power.

(b) 1 pt - Is the **Operator** a statistically significant predictor given all other predictors in model5? Use alpha = 0.05.

Q8b Answer:
No, the p-value for Operator2 is greater than 0.05 so Operator is not a statistically significant predictor given all other predictors in model5.

(c) 1 pt - Calculate the estimated dispersion parameter for model5 and interpret the result. Is this an overdispersed model using a threshold of 2?

[22]:
```
set.seed(100) # Do not change!
# with deviance residuals
wdf_d <- model5$df.residual
dev_d <- model5$deviance
dev_d/wdf_d
```

1.11872743805048

[23]:
```
set.seed(100) # Do not change!
# with pearson residuals
wdf_p <- model5$df.residual
dev_p <- sum(residuals(model5, type='pearson')^2)
dev_p/wdf_p
```

0.995968539649035

Q8c Answer:
Using a threshold of 2, model5 is not a overdispersed model because the dispersion parameters using both deviance and pearson residuals are less than 2

### 0.7.3 Question 9 - Research Questions (2 points)

(a) 1 pt - If model5 is an overdispersed model, how can we address overdispersion? No need for code, just explain in your own words.

Q9a Answer:
We can adjust the model by applying either the Quasi Poisson or the Negative Binomial to correct for overdispersion.

(b) 1 pt - For model5, is it appropriate to use Cook's Distance to check for outliers? No need for code, just explain in your own words.

Q9b Answer:
No, Cook's Distance is not appropriate for a generalized linear model since it is defined and computed assuming the residuals are from a multiple linear regression model.

## 0.8 Part 2

## 0.9 Fraud Detection in Online Transactions

This dataset simulates online transaction records, predicting whether a transaction is fraudulent or not (binary response). The binary response is without replications.

## 0.10 Features:

**Transaction_Amount (Numerical):** Transaction amount in USD ($5-10,000)

**Transaction_Hour (Numerical):** Hour of the day when the transaction occurred (0-23)

**Payment_Method (Categorical):** Credit Card, Debit Card, PayPal, Crypto

**Device_Type (Categorical):** Mobile, Desktop, Tablet

**Location_Match (Categorical):** Yes, No (whether transaction location matches the user's registered location)

**Previous_Frauds (Numerical):** Number of previous fraudulent transactions by the user (0-5)

**Account_Age_Days (Numerical):** Age of the account in days (1-5000)

**International_Transaction (Categorical):** Yes, No

**Fraudulent (Binary Output):** Whether the transaction was fraudulent (1 = Yes, 0 = No) (Response variable)

[24]:
```
set.seed(100) # Do not change!
fraud_detection=read.csv("fraud_detection.csv",header=TRUE)
fraud_detection$Payment_Method=as.factor(fraud_detection$Payment_Method)
fraud_detection$Device_Type=as.factor(fraud_detection$Device_Type)
fraud_detection$Location_Match=as.factor(fraud_detection$Location_Match)
fraud_detection$International_Transaction=as.
 ↪factor(fraud_detection$International_Transaction)
fraud_detection$Fraudulent=as.factor(fraud_detection$Fraudulent)


#Dividing the dataset into training and testing datasets
testRows = sample(nrow(fraud_detection),0.2*nrow(fraud_detection))
testData = fraud_detection[testRows, ]
trainData = fraud_detection[-testRows, ]
row.names(trainData) <- NULL
head(trainData) #display train data

options(warn = -1)
```

A data.frame: 6 × 9

| | Transaction_Amount <dbl> | Transaction_Hour <int> | Payment_Method <fct> | Device_Type <fct> | Locati <fct> |
|---|---|---|---|---|---|
| 1 | 7321.28 | 15 | PayPal | Mobile | Yes |
| 2 | 5988.59 | 23 | Debit Card | Tablet | No |
| 3 | 1564.41 | 18 | PayPal | Desktop | No |
| 4 | 1564.17 | 7 | PayPal | Tablet | Yes |
| 5 | 585.55 | 20 | Debit Card | Tablet | No |
| 6 | 8662.43 | 16 | Crypto | Tablet | No |

### 0.10.1 Q10 Data Exploration (4.5 points)

Use the dataset "fraud_detection" for this question.

a.(1.5 points) What is the overall proportion of fraudulent vs. non-fraudulent transactions? Is the dataset imbalanced?

[25]:
```
set.seed(100) # Do not change!
library(dplyr)
# Count fraudulent vs. non-fraudulent transactions
fraud_counts <- fraud_detection %>%
  count(Fraudulent) %>%
  mutate(Percentage = (n / sum(n)) * 100)

# Display the results
print(fraud_counts)
```

```
  Fraudulent   n Percentage
1          0 752         47
2          1 848         53
```

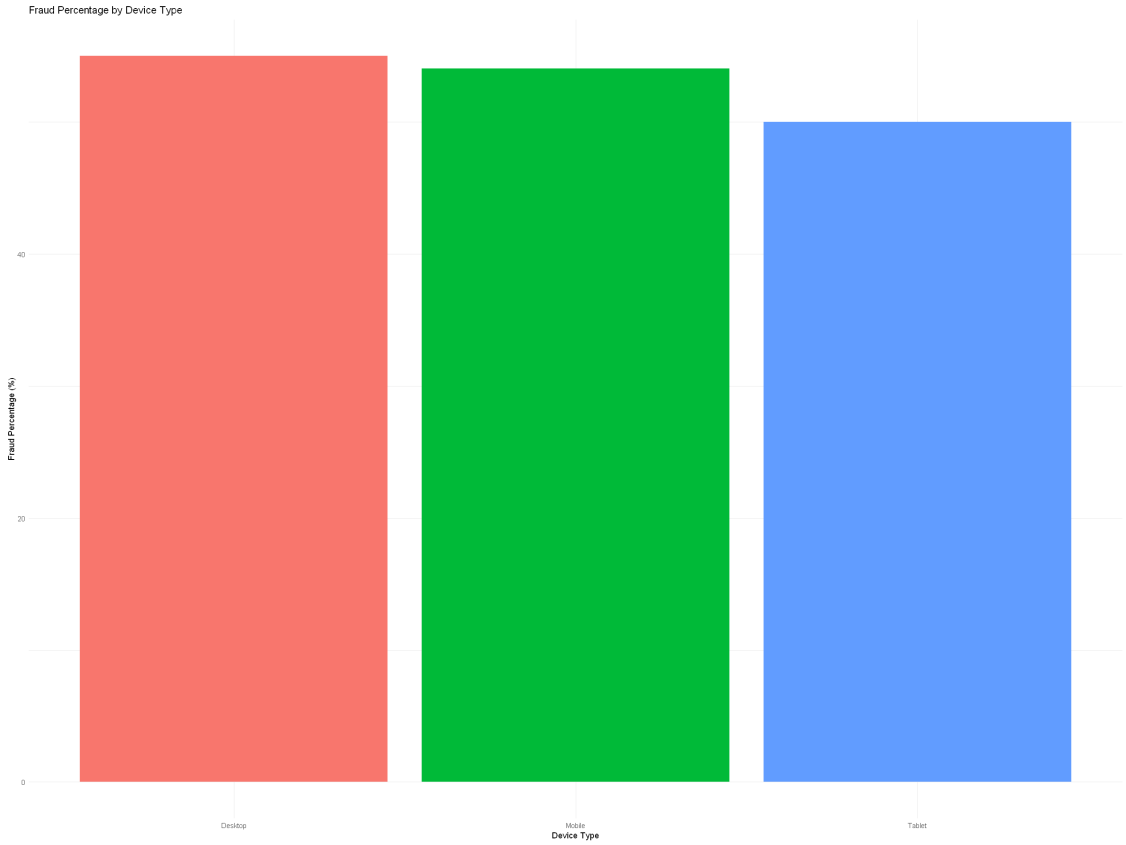
**Response to Q10a**
Yes the dataset is fairly balanced.

b. (1.5 points) Compare the percentage of fraud cases with respect to the type of device used for transaction? Use also a visual approach to compare the percentage values. Which devices are more vulnerable?

[26]:
```
set.seed(100) # Do not change!
library(dplyr)

fraud_percentage <- fraud_detection %>%
  mutate(Fraudulent = as.numeric(as.character(Fraudulent))) %>%  # Convert␣
 ↪factor to numeric
  group_by(Device_Type) %>%
  summarise(
    Total_Transactions = n(),
    Fraud_Cases = sum(Fraudulent, na.rm = TRUE),  # Ensure numeric sum
    Fraud_Percentage = (sum(Fraudulent, na.rm = TRUE) / n()) * 100
  ) %>%
  arrange(desc(Fraud_Percentage))
```

```r
print(fraud_percentage)

# Plot fraud percentage by device type
library(ggplot2)

ggplot(fraud_percentage, aes(x = reorder(Device_Type, -Fraud_Percentage), y =␣
↪Fraud_Percentage, fill = Device_Type)) +
  geom_bar(stat = "identity") +
  labs(title = "Fraud Percentage by Device Type",
       x = "Device Type",
       y = "Fraud Percentage (%)") +
  theme_minimal() +
  theme(legend.position = "none")
```

```
# A tibble: 3 x 4
  Device_Type Total_Transactions Fraud_Cases Fraud_Percentage
  <fct>                   <int>
<dbl>            <dbl>
1 Desktop                   500         275             55
2 Mobile                    568         307             54.0
3 Tablet                    532         266             50
```



**Response to Q10b**

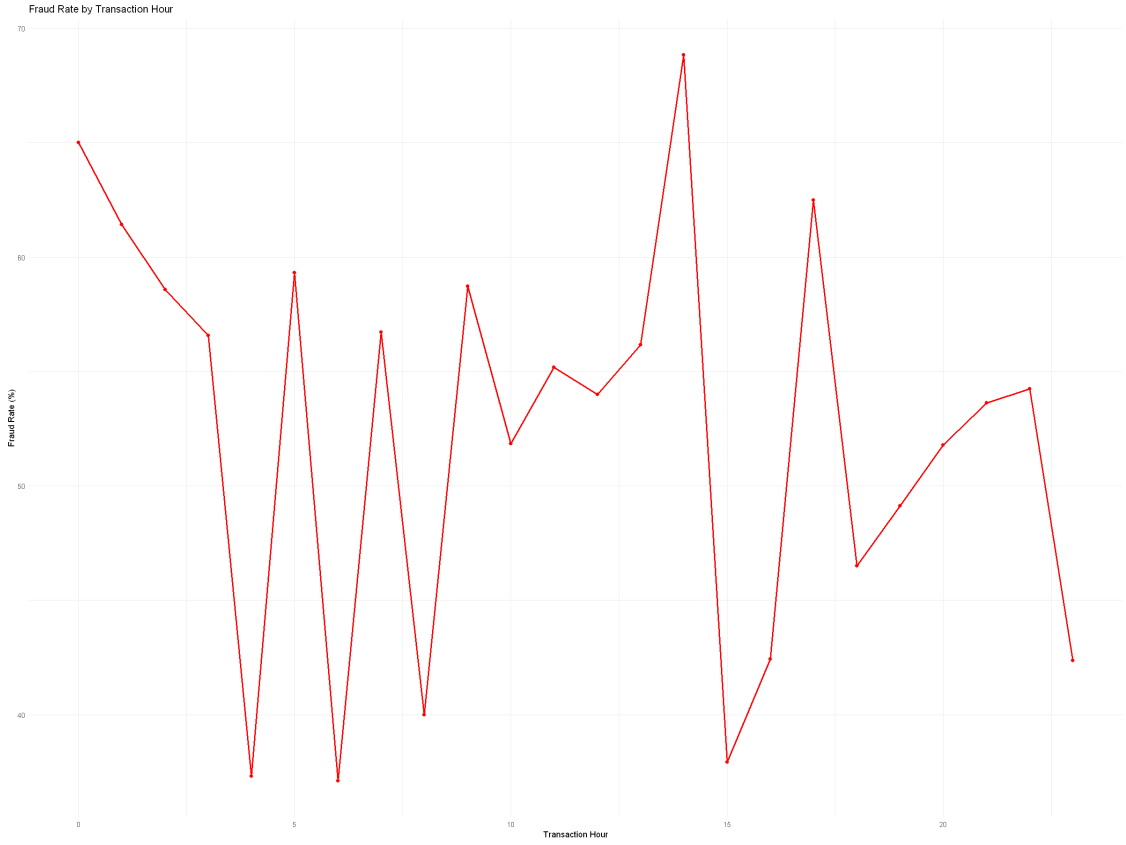
Visually looking at the plot, desktop devices have the highest proportion of fraudulent transactions with 55%. Mobile devices have the second highest proportion of fraudulent transactions with 54%. Tablet is the least vulnerable with 50% fraudulent transactions. However, it should be noted that a statistical (chi-squared) test and/or confidence intervals would be needed to be calculated to further come to a conclusion since the proportions are very close. In fact, running a proportion test, while visually it appears that Desktop and Mobile devices might be slightly more vulnerable, statistically, there is no significant difference in the proportions of fraudulent transactions among Desktop, Mobile, and Tablet devices.

c. (1.5 points) If we group transactions by Transaction_Hour, can we identify fraud-prone time windows? Also use a visual apporach for your analysis.

[27]:
```
set.seed(100) # Do not change!
# Count fraud and total transactions per hour
fraud_analysis <- fraud_detection %>%
  mutate(Fraudulent = as.numeric(as.character(Fraudulent))) %>%  # Ensure␣
 ↪numeric conversion
  group_by(Transaction_Hour) %>%
  summarise(
    Total_Transactions = n(),
    Fraud_Transactions = sum(Fraudulent, na.rm = TRUE),  # Sum fraud cases
    Fraud_Rate = (sum(Fraudulent, na.rm = TRUE) / n()) * 100  # Convert to␣
 ↪percentage
  ) %>%
  arrange(desc(Fraud_Rate))  # Sort by highest fraud rate
print(fraud_analysis)

# Plot fraud rate by transaction hour
ggplot(fraud_analysis, aes(x = Transaction_Hour, y = Fraud_Rate)) +
  geom_line(color = "red", size = 1) +
  geom_point(color = "red", size = 2) +
  labs(title = "Fraud Rate by Transaction Hour",
       x = "Transaction Hour",
       y = "Fraud Rate (%)") +
  theme_minimal()
```

```
# A tibble: 24 x 4
   Transaction_Hour Total_Transactions Fraud_Transactions Fraud_Rate
              <int>              <int>
<dbl>      <dbl>
 1               14                 77                  53       68.8
 2                0                 60                  39       65
 3               17                 88                  55       62.5
 4                1                 70                  43       61.4
 5                5                 59                  35       59.3
 6                9                 63                  37       58.7
```

```
 7              2              70              41        58.6
 8              7              67              38        56.7
 9              3              76              43        56.6
10             13              73              41        56.2
# i 14 more rows
```




**Response to Q10c**
Peak Fraud Hours:

14:00 (2 PM): 68.8% fraud rate (highest) 00:00 (12 AM): 65% fraud rate 17:00 (5 PM): 62.5% fraud rate 01:00 (1 AM): 61.4% fraud rate These hours seem to be fraud-prone, with the afternoon (2 PM) and midnight showing the highest fraud rates. This might indicate that fraudsters are most active during these times.

### 0.10.2 Q11. Logistic Regression Model (Use trainData for this question) (5 points)

a. i) (1 point) Fit a logistic regression model using "Fraudulent" as response variable and "Account_Age_Days" and "International_Transaction" as predicting variables. Call it *model12*. Display the summary of the model.

[28]:
```
set.seed(100) # Do not change!
#Code
```

```
model12=glm(Fraudulent~Account_Age_Days+International_Transaction,data=trainData,family='binom
summary(model12)
```

```
Call:
glm(formula = Fraudulent ~ Account_Age_Days + International_Transaction,
    family = "binomial", data = trainData)

Deviance Residuals:
   Min      1Q  Median      3Q     Max
-1.303  -1.198   1.059   1.156   1.167

Coefficients:
                             Estimate Std. Error z value Pr(>|z|)
(Intercept)                 2.363e-02  1.277e-01   0.185   0.8532
Account_Age_Days            5.464e-06  3.911e-05   0.140   0.8889
International_TransactionYes 2.391e-01  1.125e-01   2.125   0.0336 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1766.6  on 1279  degrees of freedom
Residual deviance: 1762.1  on 1277  degrees of freedom
AIC: 1768.1

Number of Fisher Scoring iterations: 3
```

ii) (1 point) Interpret the coefficient of "Account_Age_Days" for model12 with respect to the log-odds and odds of the response.

**Response to Q11aii**

For a day increase in the account age, the log odds of a transaction being fraudulent increases by 5.464e-06 and the odds of a transaction being fraudulent change by a factor of exp(5.464e-06) = 1.000 holding all other variables constant.

iii) (1 point) What does the value of intercept represent in terms of baseline fraud probability?

**Response to Q11aiii**

The intercept in a logistic regression model represents the log-odds of the dependent variable (fraud) when all predictor variables are zero. In this case: log-odds of fraud= 0.02363 To convert this to a baseline probability, we first compute the odds using: Odds= e^(0.02363) Odds=1.024 The probability is given by : P(Fraud)=Odds/(1+Odds) =1.024/(1.024+1)=0.5 When Account_Age_Days = 0 and International_Transaction = No, the predicted baseline probability of fraud is approximately 50%. This suggests that fraud is almost as likely as non-fraud in the absence of additional predictive factors. Since the intercept is close to zero, the baseline odds are close to 1:1, meaning fraud and non-fraud cases are nearly balanced in this scenario.

b i. (1 point) Fit a logistic regression model using "Fraudulent" as response variable and the

following predictors in the "trainData" Payment_Method, Device_Type, Location_Match, International_Transaction Call it *model22* and display the summary of model22.

[29]:
```
set.seed(100) # Do not change!
model22=glm(Fraudulent~Payment_Method+Device_Type+Location_Match+International_Transaction,dat
  ↪family='binomial')
summary(model22)
```

```
Call:
glm(formula = Fraudulent ~ Payment_Method + Device_Type + Location_Match +
    International_Transaction, family = "binomial", data = trainData)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.4576  -1.2093   0.9422   1.1110   1.3358

Coefficients:
                             Estimate Std. Error z value Pr(>|z|)
(Intercept)                 -0.130848   0.163197  -0.802  0.42268
Payment_MethodCrypto         0.317575   0.158523   2.003  0.04514 *
Payment_MethodDebit Card     0.507372   0.158458   3.202  0.00137 **
Payment_MethodPayPal         0.065831   0.164779   0.400  0.68952
Device_TypeMobile           -0.003328   0.139616  -0.024  0.98098
Device_TypeTablet           -0.234089   0.141933  -1.649  0.09909 .
Location_MatchYes            0.056034   0.113924   0.492  0.62282
International_TransactionYes 0.205562   0.113921   1.804  0.07117 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1766.6  on 1279  degrees of freedom
Residual deviance: 1744.9  on 1272  degrees of freedom
AIC: 1760.9

Number of Fisher Scoring iterations: 4
```

ii. (1 point) What does the summary of model22 suggest about the likelihood of fraud for international transactions compared to domestic transactions?

**Response to Q11bii**

The coefficient for International_TransactionYes is 0.2056, which suggests that international transactions are associated with a higher likelihood of fraud compared to domestic transactions. To interpret this in terms of odds: The odds ratio for an international transaction compared to a domestic one is given by ^0.2056 1.23 This means that the odds of a fraudulent transaction are about 23% higher for international transactions than for domestic ones.

### 0.10.3 Q12. Goodness of fit tests (Use trainData for this question) (6 points)

a.i (0.5 points) State why we cannot perform goodness-of-fit (GOF) tests for model22. a.ii. (0.5 points) Describe how we can modify the dataset to enable goodness-of-fit testing. Provide a general approach. a.iii. (2.5 points) Convert the dataset accordingly and fit a logistic regression model. Name it 'model32', and display its summary. Note: Use the same predictors as used for model22.

**Response to Q12a.i**
We cannot perform goodness-of-fit tests for model22. For goodness of fit tests, we need residuals. We can only define residuals for binary data with replications. The dataset we have is without replications.

**Response to Q12a.ii**
To perform goodness of fit tests, we have to convert the given dataset without replications into a dataset with replications.

[30]:
```
set.seed(100) # Do not change!
# Aggregate count of occurrences for each group
replicate_data.n <- aggregate(as.numeric(as.character(trainData$Fraudulent)) ~␣
 ↪Payment_Method + Device_Type + Location_Match +
                              International_Transaction,
                              data = trainData, FUN = length)

# Aggregate sum of Fraudulent transactions for each group
replicate_data.y <- aggregate(as.numeric(as.character(trainData$Fraudulent)) ~␣
 ↪Payment_Method + Device_Type + Location_Match +
                              International_Transaction,
                              data = trainData, FUN = sum)

# Merge the two datasets properly
replicate_data <- merge(replicate_data.n, replicate_data.y,
                        by = c( "Payment_Method", "Device_Type",␣
 ↪"Location_Match",
                                "International_Transaction"),
                        all = TRUE)

# Rename columns for clarity
colnames(replicate_data) <- c("Payment_Method","Device_Type", "Location_Match",␣
 ↪"International_Transaction",
                              "Total", "Fraudulent")

# View first few rows
head(replicate_data)
```

A data.frame: 6 × 6

| | Payment_Method <fct> | Device_Type <fct> | Location_Match <fct> | International_Transaction <fct> | Tot <ir |
|---|---|---|---|---|---|
| 1 | Credit Card | Desktop | No | No | 29 |
| 2 | Credit Card | Desktop | No | Yes | 14 |
| 3 | Credit Card | Desktop | Yes | No | 19 |
| 4 | Credit Card | Desktop | Yes | Yes | 24 |
| 5 | Credit Card | Mobile | No | No | 20 |
| 6 | Credit Card | Mobile | No | Yes | 20 |

[31]:
```
set.seed(100) # Do not change!
model32 <- glm(cbind(Fraudulent, Total-Fraudulent) ~ Payment_Method +␣
 ↪Device_Type + Location_Match + International_Transaction,
                family = 'binomial', data = replicate_data)
summary(model32)
```

```
Call:
glm(formula = cbind(Fraudulent, Total - Fraudulent) ~ Payment_Method +
    Device_Type + Location_Match + International_Transaction,
    family = "binomial", data = replicate_data)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.9341  -0.4427  -0.0781   0.4609   3.3582

Coefficients:
                            Estimate Std. Error z value Pr(>|z|)
(Intercept)                -0.130848   0.163196  -0.802  0.42268
Payment_MethodCrypto        0.317575   0.158523   2.003  0.04514 *
Payment_MethodDebit Card    0.507372   0.158457   3.202  0.00137 **
Payment_MethodPayPal        0.065831   0.164779   0.400  0.68952
Device_TypeMobile          -0.003328   0.139616  -0.024  0.98098
Device_TypeTablet          -0.234089   0.141933  -1.649  0.09909 .
Location_MatchYes           0.056034   0.113923   0.492  0.62282
International_TransactionYes  0.205562   0.113921   1.804  0.07116 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 80.630  on 47  degrees of freedom
Residual deviance: 58.915  on 40  degrees of freedom
AIC: 248.92

Number of Fisher Scoring iterations: 3
```

b. (1 points) Use the Deviance residuals to form goodness-of-fit hypothesis tests on model32. What do you conclude from the result of the test?

[32]:
```
set.seed(100) # Do not change!
# Deviance residuals test

cat("p-value of Deviance residuals test is:",
1-pchisq(model32$deviance, model32$df.residual), end="\n")
```

```
p-value of Deviance residuals test is: 0.02723176
```

**Response to Q12b**
According to deviance residuals , the model is not a good fit.

c. (1.5 points) Compare the summary of model22 and model32. What changes do you observe? Explain.

**Response to Q12c**
The coefficeints and their p-values are the same. The null deviance , residual deviance, degrees of freedom and AIC are different for both models. Model 32's lower null and residual deviance suggest an improved model fit, but statistical testing is required to confirm its significance. The reduction in df in Model 32 affects the GOF test, changing how we interpret statistical significance.

### 0.10.4 Q13. Decision Tree and Random Forest Models (Use trainData for this question) (4 points)

Fit the following classification models below using all the predictors in "trainData" and "Fraudulent" as the response variable. Use 3-fold cross-validation for this question.

i) (2 points) Decision Tree Model (call it *model42*).

ii) (2 points) Random Forest model (call it *model52*).

Display the summary of both models and state the average accuracy and average sensitivity for both resampled models. Which model performed better in terms of mean accuracy?

[33]:
```
#Decision Tree
set.seed(100) # Do not change!
library(caret)
library(pROC)


# Custom summary function for caret
mySummary <- function(data, lev = NULL, model = NULL) {
  # Compute the confusion matrix
  cm <- confusionMatrix(data[, "pred"], data[, "obs"],positive='Yes')

  # Compute the metrics
  out <- c(
    Accuracy = cm$overall["Accuracy"],
    Kappa = cm$overall["Kappa"],
    Sensitivity = cm$byClass["Sensitivity"],
    Specificity = cm$byClass["Specificity"],
    Pos_Pred_Value = cm$byClass["Pos Pred Value"],
```

```
    Neg_Pred_Value = cm$byClass["Neg Pred Value"],
    Precision = cm$byClass["Pos Pred Value"],
    Recall = cm$byClass["Sensitivity"],
    F1 = 2 * ((cm$byClass["Pos Pred Value"] * cm$byClass["Sensitivity"]) /
              (cm$byClass["Pos Pred Value"] + cm$byClass["Sensitivity"]))
  )

  # Compute AUC if probability column for positive class ("Yes") exists
  if ("Yes" %in% colnames(data)) {
    roc_obj <- pROC::roc(data$obs, data[["Yes"]])  # Ensure "Yes" is used␣
 ↪correctly
    out["ROC"] <- pROC::auc(roc_obj)  # Use "ROC" instead of "AUC" for caret
  }

  return(out)
}

# Ensure target variable is a factor with valid class names
trainData$Fraudulent <- factor(trainData$Fraudulent, labels = c("No", "Yes"))

# Define train control
train_ctrl <- trainControl(
  method = "cv",
  number = 3,
  classProbs = TRUE,  # Ensure probability estimation
  summaryFunction = mySummary,  # Use custom summary function
  savePredictions = "all"
)

# Train the model
model42 <- caret::train(
  Fraudulent ~ .,
  data = trainData,
  method = "rpart",
  trControl = train_ctrl,
  metric = "Accuracy"
)

# Print model details
print(model42)
print(model42$finalModel)

print("Average accuracy for decision tree:")
print(mean(model42$resample$Accuracy))

print("Average sensitivity for decision tree:")
print(mean(model42$resample$Sensitivity))
```

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

Attaching package: 'pROC'

The following objects are masked from 'package:stats':

    cov, smooth, var

Setting levels: control = No, case = Yes
Setting direction: controls > cases
Setting levels: control = No, case = Yes
Setting direction: controls < cases
Setting levels: control = No, case = Yes
Setting direction: controls < cases
Setting levels: control = No, case = Yes
Setting direction: controls < cases
Setting levels: control = No, case = Yes
Setting direction: controls < cases
Setting levels: control = No, case = Yes
Setting direction: controls < cases
Setting levels: control = No, case = Yes
Setting direction: controls < cases
Setting levels: control = No, case = Yes
Setting direction: controls < cases
Setting levels: control = No, case = Yes
Setting direction: controls < cases
Setting levels: control = No, case = Yes
```

```
Setting direction: controls < cases


CART

1280 samples
   8 predictor
   2 classes: 'No', 'Yes'

No pre-processing
Resampling: Cross-Validated (3 fold)
Summary of sample sizes: 853, 854, 853
Resampling results across tuning parameters:

  cp          Accuracy.Accuracy  Kappa.Kappa  Sensitivity.Sensitivity
  0.01355932  0.5453083          0.06773046   0.6869565
  0.02203390  0.5195417          0.02353445   0.6130435
  0.02966102  0.5335968          0.01860093   0.8231884
  Specificity.Specificity  Pos_Pred_Value.Pos Pred Value
  0.3794330                0.5647634
  0.4101057                0.5480060
  0.1949049                0.5455312
  Neg_Pred_Value.Neg Pred Value  Precision.Pos Pred Value  Recall.Sensitivity
  0.5142717                      0.5647634                 0.6869565
  0.4770034                      0.5480060                 0.6130435
  0.4874074                      0.5455312                 0.8231884
  F1.Pos Pred Value  ROC
  0.6140432          0.5292675
  0.5777806          0.5110316
  0.6503269          0.5086935

Accuracy.Accuracy was used to select the optimal model using the largest value.
The final value used for the model was cp = 0.01355932.
n= 1280

node), split, n, loss, yval, (yprob)
      * denotes terminal node

 1) root 1280 590 Yes (0.4609375 0.5390625)
   2) Transaction_Amount< 114.11 10   0 No (1.0000000 0.0000000) *
   3) Transaction_Amount>=114.11 1270 580 Yes (0.4566929 0.5433071)
     6) Account_Age_Days< 903 219  97 No (0.5570776 0.4429224)
      12) Transaction_Amount>=1076.91 194  78 No (0.5979381 0.4020619) *
      13) Transaction_Amount< 1076.91 25   6 Yes (0.2400000 0.7600000) *
     7) Account_Age_Days>=903 1051 458 Yes (0.4357755 0.5642245) *
[1] "Average accuracy for decision tree:"
[1] 0.5453083
```

```
[1] "Average sensitivity for decision tree:"
[1] 0.6869565
```

[34]:
```r
#Random forest model
set.seed(100)
# Custom summary function for caret
mySummary <- function(data, lev = NULL, model = NULL) {
  # Compute the confusion matrix
  cm <- confusionMatrix(data[, "pred"], data[, "obs"],positive="Yes")

  # Compute the metrics
  out <- c(
    Accuracy = cm$overall["Accuracy"],
    Kappa = cm$overall["Kappa"],
    Sensitivity = cm$byClass["Sensitivity"],
    Specificity = cm$byClass["Specificity"],
    Pos_Pred_Value = cm$byClass["Pos Pred Value"],
    Neg_Pred_Value = cm$byClass["Neg Pred Value"],
    Precision = cm$byClass["Pos Pred Value"],
    Recall = cm$byClass["Sensitivity"],
    F1 = 2 * ((cm$byClass["Pos Pred Value"] * cm$byClass["Sensitivity"]) /
              (cm$byClass["Pos Pred Value"] + cm$byClass["Sensitivity"]))
  )

  # Compute AUC if probability column for positive class ("Yes") exists
  if ("Yes" %in% colnames(data)) {
    roc_obj <- pROC::roc(data$obs, data[["Yes"]])  # Ensure "Yes" is used␣
 ↪correctly
    out["ROC"] <- pROC::auc(roc_obj)  # Use "ROC" instead of "AUC" for caret
  }

  return(out)
}

# Ensure target variable is a factor with valid class names
trainData$Fraudulent <- factor(trainData$Fraudulent, labels = c("No", "Yes"))

# Define train control
train_ctrl <- trainControl(
  method = "cv",
  number = 3,
  classProbs = TRUE,  # Ensure probability estimation
  summaryFunction = mySummary,  # Use custom summary function
  savePredictions = "all"
)

# Train the model
```

```r
model52 <- caret::train(
  Fraudulent ~ .,
  data = trainData,
  method = "rf",
  trControl = train_ctrl,
  metric = "Accuracy"
)

  # Print model details
print(model52)
print(model52$finalModel)

print("Average accuracy for random forest:")
print(mean(model52$resample$Accuracy))

print("Average sensitivity for random forest:")
print(mean(model52$resample$Sensitivity))
```

```
Setting levels: control = No, case = Yes

Setting direction: controls > cases

Setting levels: control = No, case = Yes

Setting direction: controls < cases

Setting levels: control = No, case = Yes

Setting direction: controls < cases

Setting levels: control = No, case = Yes

Setting direction: controls < cases

Setting levels: control = No, case = Yes

Setting direction: controls < cases

Setting levels: control = No, case = Yes

Setting direction: controls < cases

Setting levels: control = No, case = Yes

Setting direction: controls < cases

Setting levels: control = No, case = Yes
```

```
Setting direction: controls < cases

Setting levels: control = No, case = Yes

Setting direction: controls < cases

Setting levels: control = No, case = Yes

Setting direction: controls < cases


Random Forest

1280 samples
   8 predictor
   2 classes: 'No', 'Yes'

No pre-processing
Resampling: Cross-Validated (3 fold)
Summary of sample sizes: 853, 854, 853
Resampling results across tuning parameters:

  mtry  Accuracy.Accuracy  Kappa.Kappa  Sensitivity.Sensitivity
   2    0.6960928          0.3796741    0.8014493
   6    0.7140493          0.4219801    0.7623188
  11    0.7132742          0.4206991    0.7579710
  Specificity.Specificity  Pos_Pred_Value.Pos Pred Value
  0.5728444                0.6869982
  0.6576108                0.7223283
  0.6609862                0.7232705
  Neg_Pred_Value.Neg Pred Value  Precision.Pos Pred Value  Recall.Sensitivity
  0.7119697                      0.6869982                 0.8014493
  0.7041030                      0.7223283                 0.7623188
  0.7005438                      0.7232705                 0.7579710
  F1.Pos Pred Value  ROC
  0.7397387          0.7856260
  0.7415748          0.8198585
  0.7401570          0.8206480

Accuracy.Accuracy was used to select the optimal model using the largest value.
The final value used for the model was mtry = 6.

Call:
 randomForest(x = x, y = y, mtry = param$mtry)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 6
```

```
        OOB estimate of  error rate: 24.22%
Confusion matrix:
     No Yes class.error
No  415 175   0.2966102
Yes 135 555   0.1956522
[1] "Average accuracy for random forest:"
[1] 0.7140493
[1] "Average sensitivity for random forest:"
[1] 0.7623188
```

**Response to Q13**
According to mean accuracy, random forest model performed better.

### 0.10.5 Q14. Prediction (8 points)

Use the "testData" for all questions in this question.

14a) (2 points) Using testData, predict the probability of a transaction being fraudulent, and output the AVERAGE of these probabilities for each of the models below:

i) model12 (question 11a)
ii) model22 (question 11b)
iii) model42 (question 13)
iv) model52(question 13)

[35]:
```
#1
set.seed(100) # Do not change!
#Reduced Logistic regression model (model12) from question 11a
predict_model12 = predict(model12, newdata = testData,type='response')
print('Reduced Logistic regression model (model12) from question 11a prediction:
  ↪')
mean(predict_model12)
```

```
[1] "Reduced Logistic regression model (model12) from question 11a prediction:"
```

0.537774751667796

[36]:
```
#2
set.seed(100) # Do not change!
#Full Logistic regression model (model22) from question 11b
predict_model22 = predict(model22, newdata = testData,type='response')
print('Full Logistic regression model (model22) from question 11b prediction:')
mean(predict_model22)
```

```
[1] "Full Logistic regression model (model22) from question 11b prediction:"
```

0.537218041950879

[37]:
```
#3
set.seed(100) # Do not change!
#Decision tree model from question 13
```

```
predict_model42 = predict(model42, newdata = testData, type="prob")
print('Decision tree model from question 13 prediction:')
mean(predict_model42[,2])
```

```
[1] "Decision tree model from question 13 prediction:"
```

0.53588542330819

[38]:
```
#4
set.seed(100) # Do not change!
#Random forest model from question 13
predict_model52 = predict(model52, newdata = testData,type='prob')
print('Random forest model model from question 13 prediction:')
mean(predict_model52[,2])
```

```
[1] "Random forest model model from question 13 prediction:"
```

0.54081875

14b) (2 points) Using the probabilities from Q14a and a threshold of 0.55 (inclusive of 0.55), obtain the classifications of a transaction being fraudulent for all four models. Sort the classification rows by the index of the dataframe from high to low. Show the last ten classification rows for all the model classifications as well as the actual response for Fraudulent of those rows.

[39]:
```
#1
set.seed(100) # Do not change!
#Reduced Logistic regression model (model12) from question 11a
predClass.model12 = ifelse(predict_model12 >= 0.55, 1, 0)

#2
#Full Logistic regression model (model22) from question 11b
predClass.model22 = ifelse(predict_model22 >= 0.55, 1, 0)

#3
#Decision tree model from question 13
predClass.model42 = ifelse(predict_model42[,2] >= 0.55, 1, 0)

#4
#Random Forest model from question 13
predClass.model52 = ifelse(predict_model52[,2] >= 0.55, 1, 0)

# Combine the classifications and actual responses
predictions_df <- data.frame(
  Actual=testData$Fraudulent,
  ReducedLogistic = predClass.model12,
  FullLogistic=predClass.model22,
  DecisionTree = predClass.model42,
  Randomforest=predClass.model52
)
```

```r
# Sort the dataframe based on row index in descending order
predictions_df <- predictions_df[order(as.
  ↪numeric(rownames(predictions_df)),decreasing=TRUE), ]

# Print the last 10 rows
print(tail(predictions_df, 10))
```

```
   Actual ReducedLogistic FullLogistic DecisionTree Randomforest
87      1               1            1            1            1
83      0               0            0            1            0
76      1               1            0            1            0
56      1               0            0            1            1
53      1               1            1            0            1
47      1               1            0            1            1
16      0               0            1            1            0
12      0               1            1            1            1
2       1               1            1            1            1
1       1               0            0            1            0
```

14c) In this question, you will compare the prediction accuracy of the four models.

i) (2 points) Using the classifications from Q14b, create a confusion matrix and output the classification evaluation metrics for all four models. Note: every row in the classifications must be used (do not use just the last ten classification rows).

ii) (2 points) Which metric measures the rate of true positives? Which model shows the highest value for this metric?

[40]:
```r
set.seed(100) # Do not change!
library(caret)

#Create function to calculate the metrics
pred_metrics = function(modelName, predClass, actualClass) {
  cat(modelName, '\n')
  conmat <- confusionMatrix(table(predClass,actualClass),positive='1')
  c(conmat$overall["Accuracy"], conmat$byClass["Sensitivity"],
    conmat$byClass["Specificity"])
}

#metrics for Reduced logistic regression model
pred_metrics("Reduced Logistic Regression Model", predClass.model12,␣
  ↪testData$Fraudulent)
#metrics for Full logistic regression model
pred_metrics("Full Logistic Regression Model", predClass.model22,␣
  ↪testData$Fraudulent)
#metrics for Decision tree model
predictions_df$DecisionTree = as.factor(predictions_df$DecisionTree)
pred_metrics("Decision tree model",predictions_df$DecisionTree,␣
  ↪testData$Fraudulent)
```

```
#metrics for Random forest model
pred_metrics("Random Forest Model",predClass.model52,  testData$Fraudulent)
```

Reduced Logistic Regression Model

**Accuracy** 0.51875 **Sensitivity** 0.5 **Specificity** 0.537037037037037

Full Logistic Regression Model

**Accuracy** 0.54375 **Sensitivity** 0.40506329113924 **Specificity** 0.679012345679012

Decision tree model

**Accuracy** 0.515625 **Sensitivity** 0.841772151898734 **Specificity** 0.197530864197531

Random Forest Model

**Accuracy** 0.771875 **Sensitivity** 0.772151898734177 **Specificity** 0.771604938271605

**Alternative way: manually calculating accuracy, sensitivity and specificity (the method is shown for only model22, same method can be used for other models**

[41]:
```
set.seed(100) # Do not change!
# Calculate True Positives (TP)
TP <- sum(testData$Fraudulent == 1 & predClass.model22 == 1)

# Calculate True Negatives (TN)
TN <- sum(testData$Fraudulent  == 0 & predClass.model22 == 0)

# Calculate False Positives (FP)
FP <- sum(testData$Fraudulent  == 0 & predClass.model22 == 1)

# Calculate False Negatives (FN)
FN <- sum(testData$Fraudulent  == 1 & predClass.model22 == 0)

# Compute accuracy, sensitivity (recall), and specificity
accuracy = (TP + TN) / (TP + TN + FP + FN)
sensitivity = TP / (TP + FN)  # Sensitivity = Recall
specificity = TN / (TN + FP)  # Specificity

#metrics for Full logistic regression model
print("Accuracy of model22")
print(accuracy)

print("Sensitivity of model22")
print(sensitivity)

print("Specificity of model22")
print(specificity)
```

```
[1] "Accuracy of model22"
[1] 0.54375
```

```
[1] "Sensitivity of model22"
[1] 0.4050633
[1] "Specificity of model22"
[1] 0.6790123
```

**Response to Q14c**
Sensitivity, also known as recall or the true positive rate (TPR), measures how well a classification model identifies positive cases correctly. Decision tree shows the highest value for sensitivity.

# 1 Poisson Regession

We will use the dataset "poisson_data" for this question

## 1.1 Features:

**Transaction_Hour (Numerical):** Hour of the day when the transaction occurred (0-23)

**Previous_Frauds (Numerical):** Number of previous fraudulent transactions by the user (0-5)

**Account_Age_Days (Numerical):** Age of the account in days (1-5000)

**Fraud_Count (Numerical):** Number of frauds (Response variable)

### 1.1.1 Q15 Poisson Regression (Use poisson_data for this question) (2.5 points)

[42]:
```
set.seed(100) # Do not change!
poisson_data=read.csv("poisson_data.csv",header=TRUE)
head(poisson_data)
```

A data.frame: 6 × 4

| | Transaction_Hour <int> | Previous_Frauds <int> | Account_Age_Days <int> | Fraud_Count <int> |
|---|---|---|---|---|
| 1 | 13 | 3 | 1861 | 3 |
| 2 | 2 | 3 | 574 | 6 |
| 3 | 2 | 2 | 2442 | 6 |
| 4 | 6 | 1 | 1308 | 4 |
| 5 | 17 | 1 | 3584 | 1 |
| 6 | 19 | 0 | 832 | 3 |

a. i) (1 point) Fit a poisson regression model using all the predictors from the "poisson_data" and "Fraud_Count" as the response variable. Call it 'pois_model1 and display the model summary.

ii) (0.5 points) Interpret the coefficient of "Previous_Frauds" in pois_model1 with respect to the log expected "Fraud_Count".

[43]:
```
set.seed(100) # Do not change!
pois_model1 = glm(Fraud_Count~., data=poisson_data, family="poisson")
summary(pois_model1)
```

```
Call:
```

```
glm(formula = Fraud_Count ~ ., family = "poisson", data = poisson_data)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-2.52797  -0.65614  -0.02655   0.56729   2.96855

Coefficients:
                   Estimate Std. Error z value Pr(>|z|)
(Intercept)       1.178e+00  5.255e-02  22.422   <2e-16 ***
Transaction_Hour -4.651e-03  2.609e-03  -1.783   0.0746 .
Previous_Frauds   7.491e-04  1.353e-02   0.055   0.9558
Account_Age_Days -1.712e-05  1.695e-05  -1.010   0.3126
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 1070.0  on 999  degrees of freedom
Residual deviance: 1065.7  on 996  degrees of freedom
AIC: 3841.7

Number of Fisher Scoring iterations: 5
```

**Response to Q15aii**

For one unit increase in the number of previous frauds, the log expected fraud count increase by 7.491e-04 holding everything else constant.

b. (1 point) Calculate the estimated dispersion parameter for "pois_model1" using both the deviance and Pearson residuals. Is this an overdispersed model using a threshold of 2.0? Justify your answer.

[44]:
```r
set.seed(100) # Do not change!
# overdispersion with deviance residuals
wdf_d <- pois_model1$df.residual # n-p-1
dev_d <- pois_model1$deviance
overdisp_deviance = dev_d/wdf_d
print("Overdispersion with deviance residuals: ")
print(overdisp_deviance)

# overdispersion with pearson residuals
wdf_p <- pois_model1$df.residual # n-p-1
dev_p <- sum(residuals(pois_model1, type='pearson')^2)
overdisp_pearson = dev_p/wdf_p
print("Overdispersion with pearson residuals: ")
print(overdisp_pearson)
```

```
[1] "Overdispersion with deviance residuals: "
[1] 1.070005
```

```
[1] "Overdispersion with pearson residuals: "
[1] 0.96667
```

**Response to Q15b**
As the value for both deviance and pearson overdispersions are smaller than the threshold of 2.0, we conclude that overdispersion is not present.

**End of Homework**