Advance Analytics and Machine Learning

Muhammad Mattin Sharif

2024-03-15

Introduction - A1

In this report, we embark on a detailed exploration of predictive analytics, with a specific focus on determining the significant predictors that influence our target outcome. The foundation of our analysis is a meticulously curated dataset, tailored to meet the stringent requirements laid out for this project. During the initial phase of data generation, a strategic decision was made to adjust our indexing parameters, shifting the start index from 30 to 3 and extending the end index from 500 to 800. These alterations were pivotal in capturing a more comprehensive and relevant dataset, which in turn facilitated a more robust analysis.

With the data in hand, we conducted a thorough exploratory data analysis to gain initial insights and to set the stage for the subsequent application of a LASSO model. This model was instrumental in feature selection, helping us to distill the dataset to its most impactful variables. Leveraging these features, we built and validated a logistic regression model, which stood out for its interpretability and its adherence to the statistical assumptions underlying the modeling process.

The goal of this report is to not only detail the steps taken throughout this analytical journey but also to articulate the findings in a manner that aligns with professional and academic standards. From data generation to model validation, each step has been executed with precision and strategic intent, ensuring that the conclusions drawn are both robust and relevant to the broader objectives of this project.

```r
# Load Data
df <- read.csv("output_40418446.csv")
```

Statistical Summary

The statistical summary of the data was run and observations are as following:

```r
# check dimension of the dataframe
dim(df)

## [1] 1462811      77

# summary of all variables
summary(df)
```

```
##  ID_non_uniq          date_event
last_year_all_product_codes_num_uniq
##  Length:1462811     Length:1462811     Min.   :0.000
##  Class :character   Class :character   1st Qu.:1.000
##  Mode  :character   Mode  :character   Median :2.000
##                                        Mean   :1.715
##                                        3rd Qu.:2.000
##                                        Max.   :5.000
##  last_year_all_product_codes_most_freq last_year_brand_name_num_uniq
##  Min.   :   0                          Min.   :0.000
##  1st Qu.:2682                          1st Qu.:3.000
##  Median :2688                          Median :4.000
##  Mean   :2838                          Mean   :3.568
##  3rd Qu.:2699                          3rd Qu.:4.000
##  Max.   :6593                          Max.   :7.000
##  last_year_brand_name_most_freq last_year_classification0_num_uniq
##  Min.   :   0                   Min.   :0.000000
##  1st Qu.:3157                   1st Qu.:0.000000
##  Median :3240                   Median :0.000000
##  Mean   :2994                   Mean   :0.000617
##  3rd Qu.:3240                   3rd Qu.:0.000000
##  Max.   :4789                   Max.   :5.000000
##  last_year_classification1_num_uniq last_year_classification2_num_uniq
##  Min.   :    0                      Min.   :0
##  1st Qu.:  416                      1st Qu.:0
##  Median : 2482                      Median :0
##  Mean   : 2833                      Mean   :0
##  3rd Qu.: 4256                      3rd Qu.:0
##  Max.   :14716                      Max.   :0
##  last_year_company_name_num_uniq last_year_company_name_most_freq
##  Min.   :0.0000                  Min.   :  0.0
##  1st Qu.:1.0000                  1st Qu.:344.0
##  Median :1.0000                  Median :344.0
##  Mean   :0.9997                  Mean   :327.9
##  3rd Qu.:1.0000                  3rd Qu.:344.0
##  Max.   :2.0000                  Max.   :546.0
##  last_year_reason_for_legal_announcement_num_uniq
##  Min.   :0.000
##  1st Qu.:1.000
##  Median :2.000
##  Mean   :1.606
##  3rd Qu.:2.000
##  Max.   :4.000
##  last_year_reason_for_legal_announcement_most_freq
##  Min.   :   0.0
##  1st Qu.: 672.0
##  Median : 672.0
##  Mean   : 699.4
##  3rd Qu.: 711.0
##  Max.   :1263.0
##  last_year_legal_announcementing_firm_num_uniq
##  Min.   :0.000
##  1st Qu.:1.000
##  Median :1.000
##  Mean   :1.038
##  3rd Qu.:1.000
##  Max.   :3.000
##  last_year_legal_announcementing_firm_most_freq
##  Min.   :  0.0
##  1st Qu.:289.0
##  Median :289.0
##  Mean   :281.8
##  3rd Qu.:289.0
##  Max.   :474.0
##  last_year_root_cause_description_num_uniq
##  Min.   :0.000
##  1st Qu.:1.000
##  Median :1.000
##  Mean   :1.048
##  3rd Qu.:1.000
##  Max.   :4.000
##  last_year_root_cause_description_most_freq
##  Min.   : 0.000
##  1st Qu.: 4.000
##  Median : 4.000
##  Mean   : 4.522
##  3rd Qu.: 4.000
##  Max.   :42.000
##  last_year_product_quantity_average_num_uniq
##  Min.   :0.000
##  1st Qu.:1.000
##  Median :2.000
##  Mean   :1.605
##  3rd Qu.:2.000
##  Max.   :4.000
##  last_year_product_quantity_average_max
##  Min.   :     0
##  1st Qu.:559374
##  Median :559374
##  Mean   :464129
##  3rd Qu.:559374
##  Max.   :559374
##  last_year_product_quantity_average_average
##  Min.   :     0
##  1st Qu.:279730
##  Median :279730
##  Mean   :307647
##  3rd Qu.:559374
##  Max.   :559374
##  last_year_decision_date_max_changes_in_product
##  Min.   : 0.00
##  1st Qu.:15.00
##  Median :16.00
##  Mean   :16.34
##  3rd Qu.:17.00
##  Max.   :49.00
##  last_year_decision_date_average_changes_in_product
##  Min.   : 0.00
##  1st Qu.:15.00
##  Median :16.00
##  Mean   :16.34
##  3rd Qu.:17.00
##  Max.   :49.00
##  last_two_years_all_product_codes_num_uniq
##  Min.   :0.000
##  1st Qu.:1.000
##  Median :2.000
##  Mean   :1.715
##  3rd Qu.:2.000
##  Max.   :5.000
##  last_two_years_all_product_codes_most_freq
last_two_years_brand_name_num_uniq
##  Min.   :   0                              Min.   :0.000
##  1st Qu.:2682                              1st Qu.:3.000
##  Median :2688                              Median :4.000
##  Mean   :2839                              Mean   :3.568
##  3rd Qu.:2699                              3rd Qu.:4.000
##  Max.   :6593                              Max.   :7.000
##  last_two_years_brand_name_most_freq
last_two_years_classification0_num_uniq
##  Min.   :   0                       Min.   : 0.000000
##  1st Qu.:3157                       1st Qu.: 0.000000
##  Median :3240                       Median : 0.000000
##  Mean   :2994                       Mean   : 0.001051
##  3rd Qu.:3240                       3rd Qu.: 0.000000
##  Max.   :4789                       Max.   :10.000000
##  last_two_years_classification1_num_uniq
##  Min.   :    0
##  1st Qu.:  416
##  Median : 2482
##  Mean   : 2833
##  3rd Qu.: 4256
##  Max.   :14716
##  last_two_years_classification2_num_uniq
last_two_years_company_name_num_uniq
##  Min.   :0                              Min.   :0
##  1st Qu.:0                              1st Qu.:1
##  Median :0                              Median :1
##  Mean   :0                              Mean   :1
##  3rd Qu.:0                              3rd Qu.:1
##  Max.   :0                              Max.   :2
##  last_two_years_company_name_most_freq
##  Min.   :  0
##  1st Qu.:344
##  Median :344
##  Mean   :328
##  3rd Qu.:344
##  Max.   :546
##  last_two_years_reason_for_legal_announcement_num_uniq
##  Min.   :0.000
##  1st Qu.:1.000
##  Median :2.000
##  Mean   :1.634
##  3rd Qu.:2.000
##  Max.   :8.000
##  last_two_years_reason_for_legal_announcement_most_freq
##  Min.   :   0.0
##  1st Qu.: 672.0
##  Median : 672.0
##  Mean   : 701.5
##  3rd Qu.: 711.0
##  Max.   :1263.0
##  last_two_years_legal_announcementing_firm_num_uniq
##  Min.   :0.00
##  1st Qu.:1.00
##  Median :1.00
##  Mean   :1.04
##  3rd Qu.:1.00
##  Max.   :3.00
##  last_two_years_legal_announcementing_firm_most_freq
##  Min.   :  0.0
##  1st Qu.:289.0
##  Median :289.0
##  Mean   :283.2
##  3rd Qu.:289.0
##  Max.   :401.0
##  last_two_years_root_cause_description_num_uniq
##  Min.   :0.000
##  1st Qu.:1.000
##  Median :1.000
##  Mean   :1.072
##  3rd Qu.:1.000
##  Max.   :6.000
##  last_two_years_root_cause_description_most_freq
##  Min.   : 0.000
##  1st Qu.: 4.000
##  Median : 4.000
##  Mean   : 4.685
##  3rd Qu.: 4.000
##  Max.   :42.000
##  last_two_years_product_quantity_average_num_uniq
##  Min.   :0.000
##  1st Qu.:1.000
##  Median :2.000
##  Mean   :1.629
##  3rd Qu.:2.000
##  Max.   :7.000
##  last_two_years_product_quantity_average_max
##  Min.   :     0
##  1st Qu.:559374
##  Median :559374
##  Mean   :464211
##  3rd Qu.:559374
##  Max.   :559374
##  last_two_years_product_quantity_average_average
##  Min.   :     0
##  1st Qu.:279730
##  Median :279730
##  Mean   :307571
##  3rd Qu.:559374
##  Max.   :559374
##  last_two_years_decision_date_max_changes_in_product
##  Min.   : 0.00
##  1st Qu.:28.00
##  Median :29.00
##  Mean   :28.98
##  3rd Qu.:29.00
##  Max.   :91.00
##  last_two_years_decision_date_average_changes_in_product
##  Min.   : 0.00
##  1st Qu.:28.00
##  Median :29.00
##  Mean   :28.98
##  3rd Qu.:29.00
##  Max.   :91.00
##  last_four_years_all_product_codes_num_uniq
##  Min.   :1.000
##  1st Qu.:1.000
##  Median :2.000
##  Mean   :1.715
##  3rd Qu.:2.000
##  Max.   :5.000
##  last_four_years_all_product_codes_most_freq
##  Min.   :  55
##  1st Qu.:2682
##  Median :2688
##  Mean   :2839
##  3rd Qu.:2699
##  Max.   :6593
##  last_four_years_brand_name_num_uniq last_four_years_brand_name_most_freq
##  Min.   :1.000                       Min.   :   7
##  1st Qu.:3.000                       1st Qu.:3157
##  Median :4.000                       Median :3240
##  Mean   :3.568                       Mean   :2994
##  3rd Qu.:4.000                       3rd Qu.:3240
##  Max.   :7.000                       Max.   :4789
##  last_four_years_classification0_num_uniq
##  Min.   : 0.00000
##  1st Qu.: 0.00000
##  Median : 0.00000
##  Mean   : 0.00114
##  3rd Qu.: 0.00000
##  Max.   :10.00000
##  last_four_years_classification1_num_uniq
##  Min.   :    0
##  1st Qu.:  416
##  Median : 2482
##  Mean   : 2833
##  3rd Qu.: 4256
##  Max.   :14716
##  last_four_years_classification2_num_uniq
last_four_years_company_name_num_uniq
##  Min.   :0                                Min.   :1
##  1st Qu.:0                                1st Qu.:1
##  Median :0                                Median :1
##  Mean   :0                                Mean   :1
##  3rd Qu.:0                                3rd Qu.:1
##  Max.   :0                                Max.   :2
##  last_four_years_company_name_most_freq
##  Min.   :105
##  1st Qu.:344
##  Median :344
##  Mean   :328
##  3rd Qu.:344
##  Max.   :546
##  last_four_years_reason_for_legal_announcement_num_uniq
##  Min.   :1.000
##  1st Qu.:1.000
##  Median :2.000
##  Mean   :1.634
##  3rd Qu.:2.000
##  Max.   :8.000
##  last_four_years_reason_for_legal_announcement_most_freq
##  Min.   : 112.0
##  1st Qu.: 672.0
##  Median : 672.0
##  Mean   : 701.5
##  3rd Qu.: 711.0
##  Max.   :1263.0
##  last_four_years_legal_announcementing_firm_num_uniq
##  Min.   :1.00
##  1st Qu.:1.00
##  Median :1.00
##  Mean   :1.04
##  3rd Qu.:1.00
##  Max.   :3.00
##  last_four_years_legal_announcementing_firm_most_freq
##  Min.   :103.0
##  1st Qu.:289.0
##  Median :289.0
##  Mean   :283.2
##  3rd Qu.:289.0
##  Max.   :401.0
##  last_four_years_root_cause_description_num_uniq
##  Min.   :1.000
##  1st Qu.:1.000
##  Median :1.000
##  Mean   :1.072
##  3rd Qu.:1.000
##  Max.   :6.000
##  last_four_years_root_cause_description_most_freq
##  Min.   : 3.000
##  1st Qu.: 4.000
##  Median : 4.000
##  Mean   : 4.685
##  3rd Qu.: 4.000
##  Max.   :42.000
##  last_four_years_product_quantity_average_num_uniq
##  Min.   :1.00
##  1st Qu.:1.00
##  Median :2.00
##  Mean   :1.63
##  3rd Qu.:2.00
##  Max.   :7.00
##  last_four_years_product_quantity_average_max
##  Min.   :     2
##  1st Qu.:559374
##  Median :559374
##  Mean   :464211
##  3rd Qu.:559374
##  Max.   :559374
##  last_four_years_product_quantity_average_average
##  Min.   :     2
##  1st Qu.:279730
##  Median :279730
##  Mean   :307568
##  3rd Qu.:559374
##  Max.   :559374
##  last_four_years_decision_date_max_changes_in_product
##  Min.   :  1.00
##  1st Qu.: 63.00
##  Median : 64.00
##  Mean   : 59.69
##  3rd Qu.: 64.00
##  Max.   :165.00
##  last_four_years_decision_date_average_changes_in_product
##  Min.   :  1.00
##  1st Qu.: 63.00
##  Median : 64.00
##  Mean   : 59.69
##  3rd Qu.: 64.00
##  Max.   :165.00
##  Proudct.issue.consequence manufacturer_contact_address_1
product.brand_name
##  Length:1462811            Min.   :   62                  Min.   : 13250
##  Class :character          1st Qu.: 3018                  1st Qu.:215593
##  Mode  :character          Median : 3018                  Median :215593
##                            Mean   : 4073                  Mean   :197322
##                            3rd Qu.: 3018                  3rd Qu.:215795
##                            Max.   :13145                  Max.   :344588
##  product.generic_name product.issue.type type_of_report.1
reporter_job_code
##  Min.   :  1860       Min.   :  3.0      Min.   :0.000    Min.   : 8.00
##  1st Qu.: 46965       1st Qu.:399.0      1st Qu.:0.000    1st Qu.:32.00
##  Median : 46965       Median :597.0      Median :0.000    Median :36.00
##  Mean   : 52159       Mean   :502.9      Mean   :0.473    Mean   :34.93
##  3rd Qu.: 46965       3rd Qu.:599.0      3rd Qu.:1.000    3rd Qu.:36.00
##  Max.   :101028       Max.   :964.0      Max.   :1.000    Max.   :52.00
##   source_type     product.manufacturer_name product.product_operator
##  Min.   : 3.000   Min.   : 5575             Min.   : 0.00
##  1st Qu.: 6.000   1st Qu.:19285             1st Qu.:21.00
##  Median : 6.000   Median :19285             Median :21.00
##  Mean   : 8.358   Mean   :18342             Mean   :20.88
##  3rd Qu.:10.000   3rd Qu.:19408             3rd Qu.:21.00
##  Max.   :23.000   Max.   :31471             Max.   :41.00
##  product.manufacturer_city product.manufacturer_state
##  Min.   :  874             Min.   : 7.00
##  1st Qu.: 4513             1st Qu.: 8.00
##  Median : 6923             Median :32.00
##  Mean   : 6137             Mean   :30.06
##  3rd Qu.: 6923             3rd Qu.:48.00
##  Max.   :10778             Max.   :63.00
##  product.manufacturer_country product.field_description
##  Min.   : 21.0                Length:1462811
##  1st Qu.:126.0                Class :character
##  Median :126.0                Mode  :character
##  Mean   :125.8
##  3rd Qu.:126.0
##  Max.   :135.0
##  product.product_report_product_code
##  Length:1462811
##  Class :character
##  Mode  :character
##
##
##
```

Overall, dataset is pretty large, comprising 1,462,811 entries across 77 different attributes. It encapsulates a wide array of information primarily focused on product data, brand names, legal announcements, and various classification details.

Data Exploration and Preprocessing - A2

In the following analysis, we delve into our dataset to uncover patterns and insights that lie beneath the surface. Specifically, we focus on identifying and understanding the prevalence of zero values across different variables, the occurrence of 'UNKNOWN' values within the 'product.field description', and categorically analyzing the distribution of zeros across time-based variables. This examination aims to provide a comprehensive understanding of the dataset's characteristics, aiding in more informed decision-making for subsequent analyses.

Our first step is to assess the prevalence of zero values across all variables in the dataset. This initial analysis helps us gauge the extent of data that might be considered 'inactive' or 'unused' within our dataset. By identifying variables with a high count of zeros, we can better understand which features may require further investigation or pre-processing.

```
# Count zeros in each column
zero_count <- sapply(df, function(x) sum(x == 0))

# Create a new data frame with variable names and their corresponding zero
counts
zero_count_df <- data.frame(
  Variable = names(zero_count),
  ZeroCount = as.numeric(zero_count)
)

# View the new data frame
zero_count_df

##                                                 Variable ZeroCount
## 1                                            ID_non_uniq         0
## 2                                             date_event         0
## 3                      last_year_all_product_codes_num_uniq       443
## 4                     last_year_all_product_codes_most_freq       443
## 5                              last_year_brand_name_num_uniq       443
## 6                             last_year_brand_name_most_freq       443
## 7                         last_year_classification0_num_uniq   1462391
## 8                         last_year_classification1_num_uniq       863
## 9                         last_year_classification2_num_uniq   1462811
## 10                           last_year_company_name_num_uniq       443
## 11                          last_year_company_name_most_freq       443
## 12           last_year_reason_for_legal_announcement_num_uniq       443
## 13          last_year_reason_for_legal_announcement_most_freq       443
## 14              last_year_legal_announcementing_firm_num_uniq       443
## 15             last_year_legal_announcementing_firm_most_freq       443
## 16                  last_year_root_cause_description_num_uniq       443
## 17                 last_year_root_cause_description_most_freq       443
## 18                last_year_product_quantity_average_num_uniq       443
## 19                     last_year_product_quantity_average_max       443
## 20                 last_year_product_quantity_average_average       443
## 21          last_year_decision_date_max_changes_in_product      2202
## 22      last_year_decision_date_average_changes_in_product      2202
## 23               last_two_years_all_product_codes_num_uniq        60
## 24              last_two_years_all_product_codes_most_freq        60
## 25                      last_two_years_brand_name_num_uniq        60
## 26                     last_two_years_brand_name_most_freq        60
## 27                 last_two_years_classification0_num_uniq   1462133
## 28                 last_two_years_classification1_num_uniq       738
## 29                 last_two_years_classification2_num_uniq   1462811
## 30                    last_two_years_company_name_num_uniq        60
## 31                   last_two_years_company_name_most_freq        60
## 32    last_two_years_reason_for_legal_announcement_num_uniq        60
## 33   last_two_years_reason_for_legal_announcement_most_freq        60
## 34       last_two_years_legal_announcementing_firm_num_uniq        60
## 35      last_two_years_legal_announcementing_firm_most_freq        60
## 36           last_two_years_root_cause_description_num_uniq        60
## 37          last_two_years_root_cause_description_most_freq        60
## 38         last_two_years_product_quantity_average_num_uniq        60
## 39              last_two_years_product_quantity_average_max        60
## 40          last_two_years_product_quantity_average_average        60
## 41      last_two_years_decision_date_max_changes_in_product      1479
## 42  last_two_years_decision_date_average_changes_in_product      1479
## 43              last_four_years_all_product_codes_num_uniq         0
## 44             last_four_years_all_product_codes_most_freq         0
## 45                     last_four_years_brand_name_num_uniq         0
## 46                    last_four_years_brand_name_most_freq         0
## 47                last_four_years_classification0_num_uniq   1462073
## 48                last_four_years_classification1_num_uniq       738
## 49                last_four_years_classification2_num_uniq   1462811
## 50                   last_four_years_company_name_num_uniq         0
## 51                  last_four_years_company_name_most_freq         0
## 52   last_four_years_reason_for_legal_announcement_num_uniq         0
## 53  last_four_years_reason_for_legal_announcement_most_freq         0
## 54      last_four_years_legal_announcementing_firm_num_uniq         0
## 55     last_four_years_legal_announcementing_firm_most_freq         0
## 56          last_four_years_root_cause_description_num_uniq         0
## 57         last_four_years_root_cause_description_most_freq         0
## 58        last_four_years_product_quantity_average_num_uniq         0
## 59             last_four_years_product_quantity_average_max         0
## 60         last_four_years_product_quantity_average_average         0
## 61     last_four_years_decision_date_max_changes_in_product         0
## 62 last_four_years_decision_date_average_changes_in_product         0
## 63                              Proudct.issue.consequence         0
## 64                          manufacturer_contact_address_1         0
## 65                                      product.brand_name         0
## 66                                    product.generic_name         0
## 67                                      product.issue.type         0
## 68                                        type_of_report.1    770925
## 69                                       reporter_job_code         0
## 70                                             source_type         0
## 71                              product.manufacturer_name         0
## 72                               product.product_operator     20241
## 73                              product.manufacturer_city         0
## 74                             product.manufacturer_state         0
## 75                           product.manufacturer_country         0
## 76                              product.field_description         0
## 77                       product.product_report_product_code      0
```

Moving beyond numerical values, we next explore the ‘product.field description’ variable to identify rows labeled as ‘Unknown’. This analysis is crucial for assessing the completeness of our dataset, as ‘Unknown’ values can indicate missing or unclassified information, impacting the accuracy of our future analyses.

```
# 1. Analyze rows with "UNKNOWN" in 'product.field description'
unknown_field_rows <- df %>% filter(product.field_description == "Unknown")

# Analyzing the count of such rows
unknown_field_count <- nrow(unknown_field_rows)
print(paste("Number of rows with 'UNKNOWN' in 'product.field description':",
unknown_field_count))

## [1] "Number of rows with 'UNKNOWN' in 'product.field description':
1383796"
```

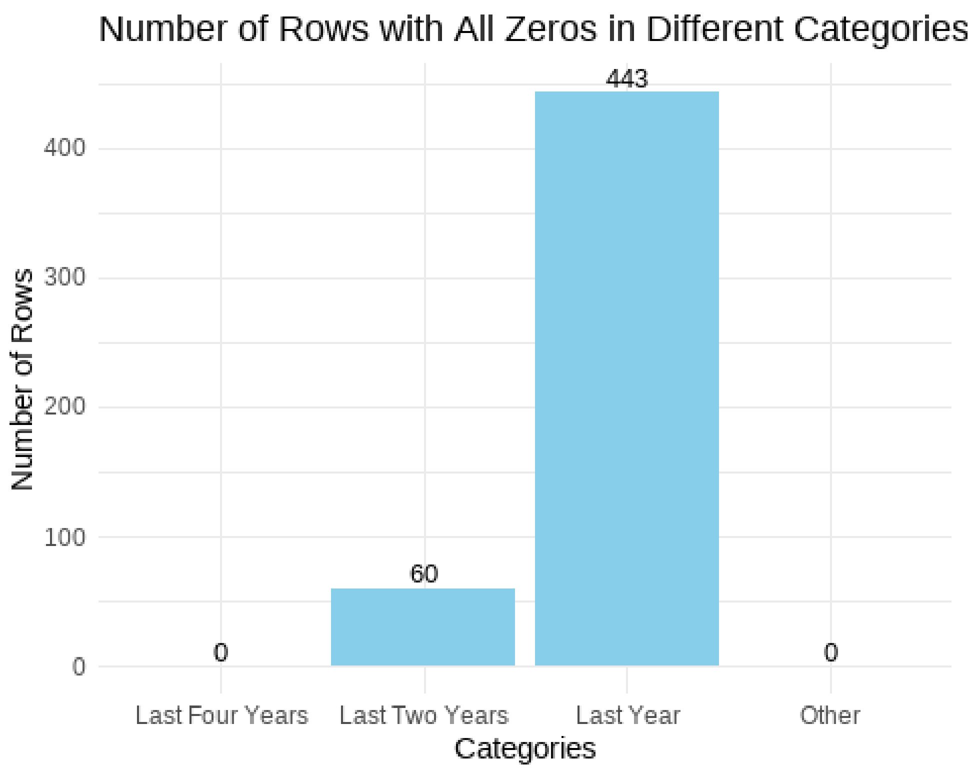
With a clear understanding of zero value prevalence and ‘Unknown’ descriptions, we now categorize our dataset based on time references—specifically, variables related to ‘last year’, ‘last two years’, and ‘last four years’, along with other variables. This categorization allows us to examine patterns of inactivity or missing information over different time frames, offering insights into the temporal dynamics of our dataset. the count is as following:

```
filter_zeros_in_category <- function(df, columns) {
  # Create a logical vector for each row, indicating if all selected columns
are zero
  rows_with_all_zeros <- apply(df[, columns], 1, function(row) all(row == 0))
  # Filter and return the dataframe with rows where all specified columns are
zero
  df[rows_with_all_zeros, ]
}

columns_last_year <- grep("last_year", names(df), value = TRUE)
columns_last_two_years <- grep("last_two_years", names(df), value = TRUE)
columns_last_four_years <- grep("last_four_years", names(df), value = TRUE)
columns_others <- setdiff(names(df), c(columns_last_year,
columns_last_two_years, columns_last_four_years))

# Filter rows for each category and store in new dataframes
df_last_year_zeros <- filter_zeros_in_category(df, columns_last_year)
df_last_two_years_zeros <- filter_zeros_in_category(df,
columns_last_two_years)
```

```r
df_last_four_years_zeros <- filter_zeros_in_category(df,
columns_last_four_years)
df_others_zeros <- filter_zeros_in_category(df, columns_others)

# Apply corrected function for each category
count_zeros_last_year <- nrow(df_last_year_zeros)
count_zeros_last_two_years <- nrow(df_last_two_years_zeros)
count_zeros_last_four_years <- nrow(df_last_four_years_zeros)
count_zeros_others <- nrow(df_others_zeros)

# Corrected output
print(paste("Rows with all zeros in 'last year' category:",
count_zeros_last_year))

## [1] "Rows with all zeros in 'last year' category: 443"

print(paste("Rows with all zeros in 'last two years' category:",
count_zeros_last_two_years))

## [1] "Rows with all zeros in 'last two years' category: 60"

print(paste("Rows with all zeros in 'last four years' category:",
count_zeros_last_four_years))

## [1] "Rows with all zeros in 'last four years' category: 0"

print(paste("Rows with all zeros in 'other' category:", count_zeros_others))

## [1] "Rows with all zeros in 'other' category: 0"
```

To visualize our findings and make them more accessible, we'll plot the count of rows with all zeros across the different time-based categories we've analyzed.

```r
# Prepare data in a data frame for ggplot
data_to_plot <- data.frame(
  Category = c("Last Year", "Last Two Years", "Last Four Years","Other"),
  Count = c(count_zeros_last_year, count_zeros_last_two_years,
count_zeros_last_four_years, count_zeros_others)
)

# Create the ggplot
ggplot(data_to_plot, aes(x = Category, y = Count, label = Count)) +
  geom_bar(stat = "identity", fill = "skyblue") +
  geom_text(vjust = -0.3, size = 3.5) + # Adjust vjust for label position,
size for font size
  labs(title = "Number of Rows with All Zeros in Different Categories",
       x = "Categories", y = "Number of Rows") +
  theme_minimal()
```



key insights:

- A significant number of recent inactivities with 443 rows having all zeros in the 'last year' category. Reduced inactivity observed over the past two years, with only 60 rows showing all zeros.
- No inactivity detected in both the 'last four years' and 'other' categories, indicating consistent activity or data recording over longer periods.
- These patterns suggest a fluctuation in activity levels, with more consistent data availability over extended timelines.

Defining Catagorical features - A3

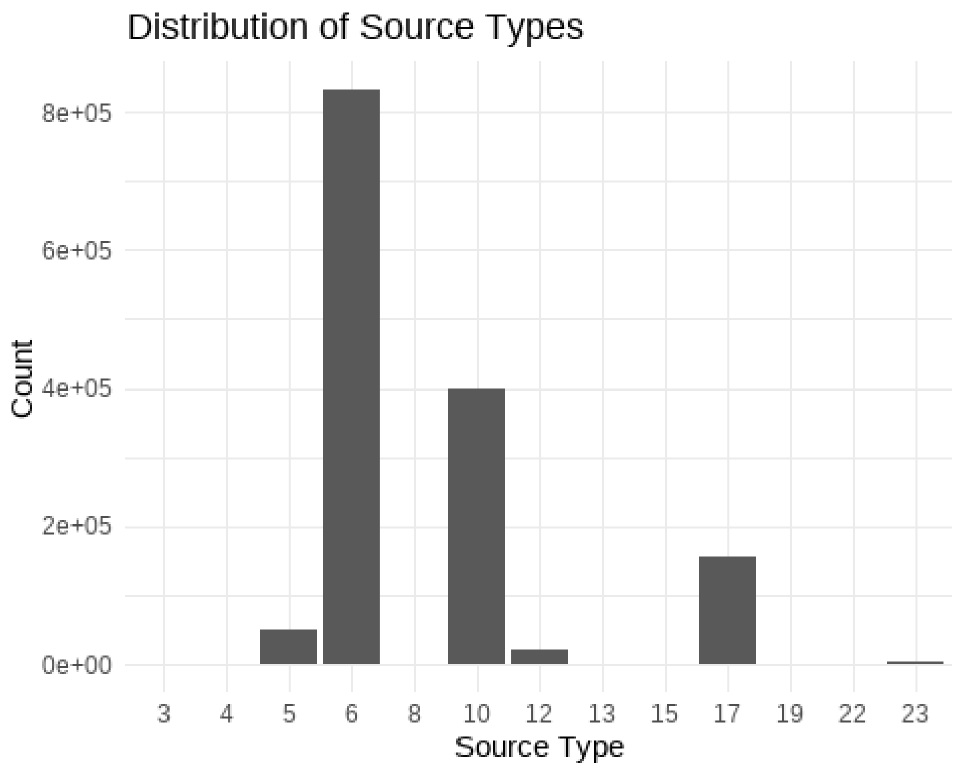
Converting multiple categorical variables in the dataframe to factor type to facilitate analysis involving categorical data. This process includes identifiers, product information, manufacturer details, and issue types etc.

```
df1 <- df%>%
  mutate_if(is.character, as.factor)

#converting categorical to factor
df1$ID_non_uniq <- as.factor(df1$ID_non_uniq)
df1$Proudct.issue.consequence <- as.factor(df1$Proudct.issue.consequence)
df1$manufacturer_contact_address_1 <-
as.factor(df1$manufacturer_contact_address_1)
df1$product.brand_name <- as.factor(df1$product.brand_name)
df1$product.generic_name <- as.factor(df1$product.generic_name)
df1$`product_issue_type` <- as.factor(df1$`product.issue.type`)
df1$type_of_report.1 <- as.factor(df1$type_of_report.1)
df1$reporter_job_code <- as.factor(df1$reporter_job_code)
df1$source_type <- as.factor(df1$source_type)
df1$product.manufacturer_name <- as.factor(df1$product.manufacturer_name)
df1$product.product_operator <- as.factor(df1$product.product_operator)
df1$product.manufacturer_city <- as.factor(df1$product.manufacturer_city)
df1$product.manufacturer_state <- as.factor(df1$product.manufacturer_state)
df1$product.manufacturer_country <-
as.factor(df1$product.manufacturer_country)
df1$product.field_description <- as.factor(df1$product.field_description)
df1$product.product_report_product_code <-
as.factor(df1$product.product_report_product_code)
```

Graphical Analysis - A4

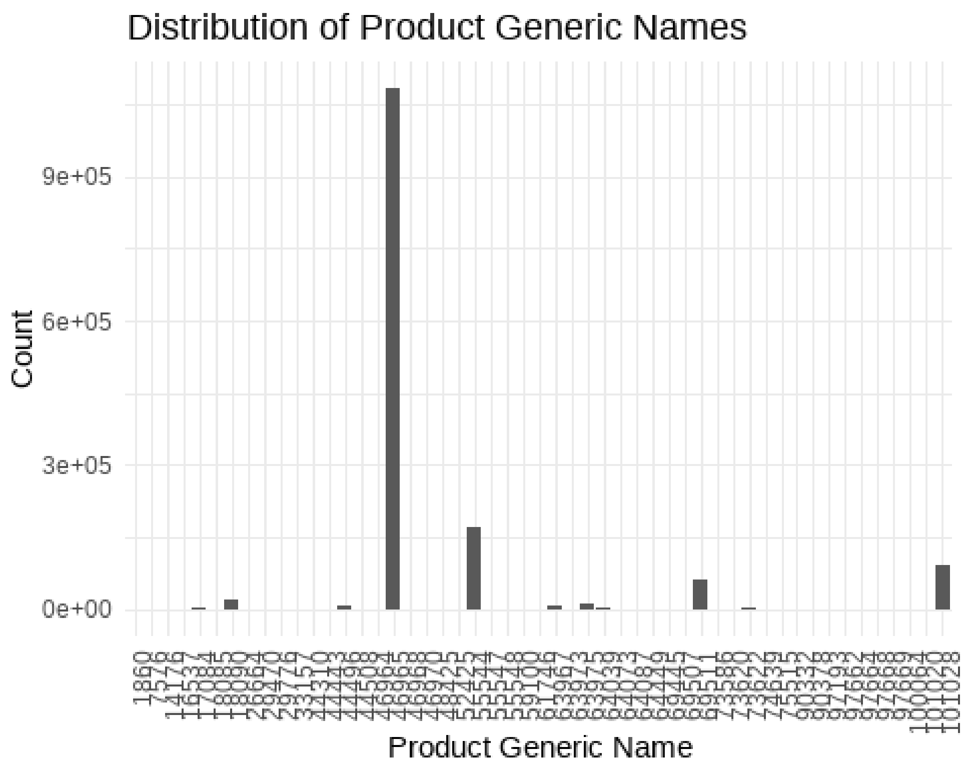
This bar plot visualizes the distribution of different source types within a dataset. Each bar represents a different source type, categorized by numerical identifiers on the x-axis (e.g., 3, 4, 5, 6, etc.). The height of each bar indicates the count of records for each source type. The plot indicates that source type 6 is the most common, followed by source type 10 and 17, with source types 3, 5, 8, 12, 15, 19, 22, and 23 being less frequent.

```r
ggplot(df1, aes(x=factor(source_type))) +
  geom_bar() +
  theme_minimal() +
  labs(title="Distribution of Source Types",
       x="Source Type",
       y="Count")
```



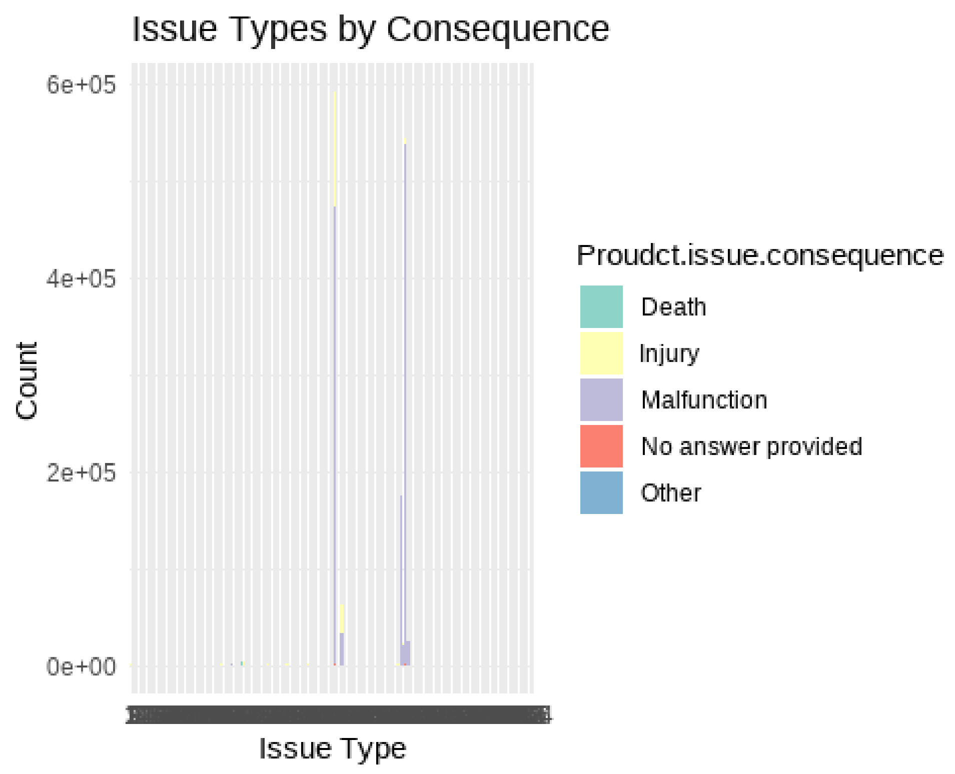
One product generic name, represented by the numerical identifier near the center of the x-axis "46964", has a significantly higher count compared to the others, indicating it is the most frequently occurring product in the dataset

```r
# Simple bar chart for product.generic_name
ggplot(df1, aes(x=factor(product.generic_name))) +
  geom_bar() +
  theme_minimal() +
  labs(title="Distribution of Product Generic Names",
       x="Product Generic Name",
       y="Count") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) # Rotate X axis
labels if they overlap
```



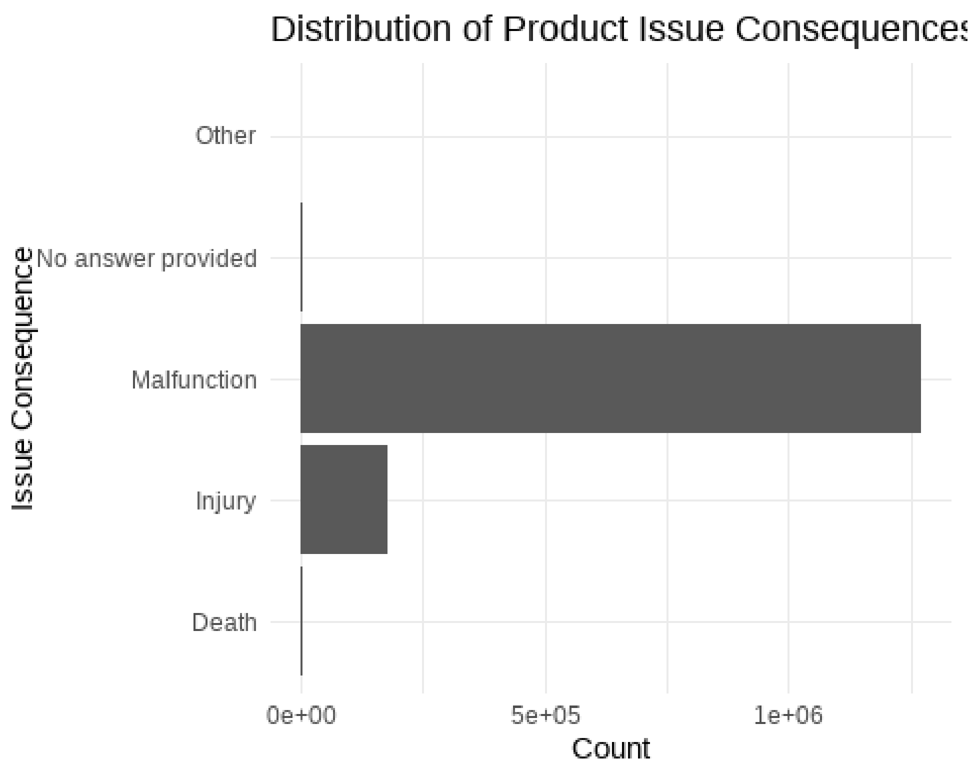
It’s notable that certain issue types have a very high count, particularly one that stands out with a significant peak, indicating a high number of reports of one specific issue type.

```
# Plotting
ggplot(df1, aes(x=factor(product.issue.type),
fill=Proudct.issue.consequence)) +
  geom_bar() +
  theme_minimal() +
  labs(title="Issue Types by Consequence",
       x="Issue Type",
       y="Count") +
  scale_fill_brewer(palette="Set3") # Use a color palette that's easy to
differentiate
```



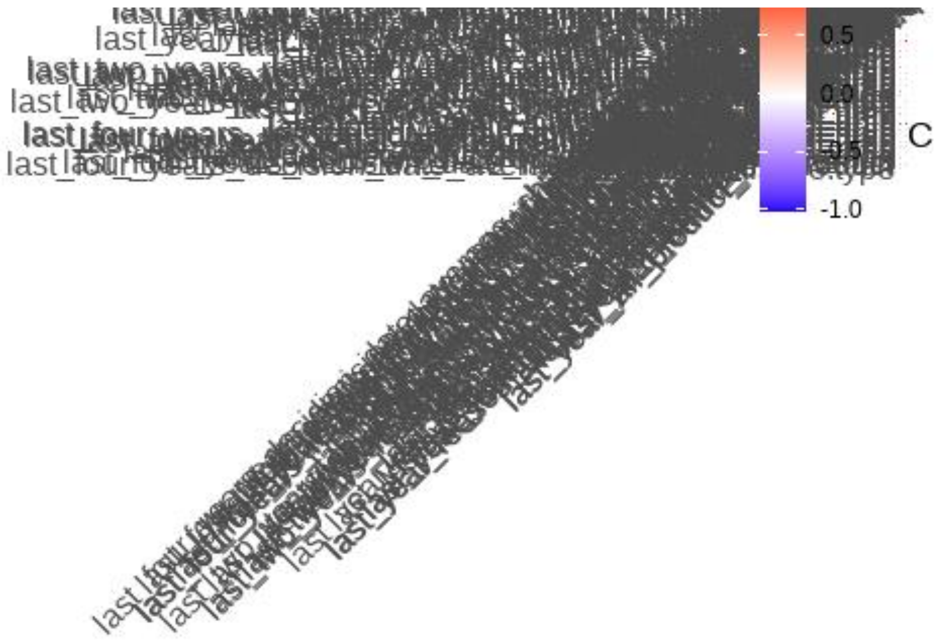
The bar for “Malfunction” is significantly longer than the others, indicating it is the most frequently reported issue consequence. “No answer provided” and “Other” also appear somewhat frequently, while “Injury” and “Death” are less common.

```
# Plotting
ggplot(df1, aes(x=Proudct.issue.consequence)) +
  geom_bar() +
  theme_minimal() +
  labs(title="Distribution of Product Issue Consequences",
       x="Issue Consequence",
       y="Count") +
  coord_flip() # Optional: Flips the axes for better readability if the
labels are long
```



Correlation Analysis between numeric features

```
# Selecting only numeric columns
numeric_data <- df1[sapply(df1, is.numeric)]

# Calculating correlation matrix
corr_matrix <- cor(numeric_data, use = "complete.obs")

## Warning in cor(numeric_data, use = "complete.obs"): the standard deviation
is
## zero

# Melting the correlation matrix
melted_corr_matrix <- melt(corr_matrix)


# Plotting the heatmap
ggplot(data = melted_corr_matrix, aes(x = Var1, y = Var2, fill = value)) +
  geom_tile(color = "white") +
  scale_fill_gradient2(low = "blue", high = "red", mid = "white", midpoint =
0, limit = c(-1,1), space = "Lab", name="Correlation") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, vjust = 1, size = 12, hjust =
1),
        axis.text.y = element_text(size = 12)) +
  labs(x = "", y = "", title = "Correlation Matrix Heatmap") +
  coord_fixed()
```



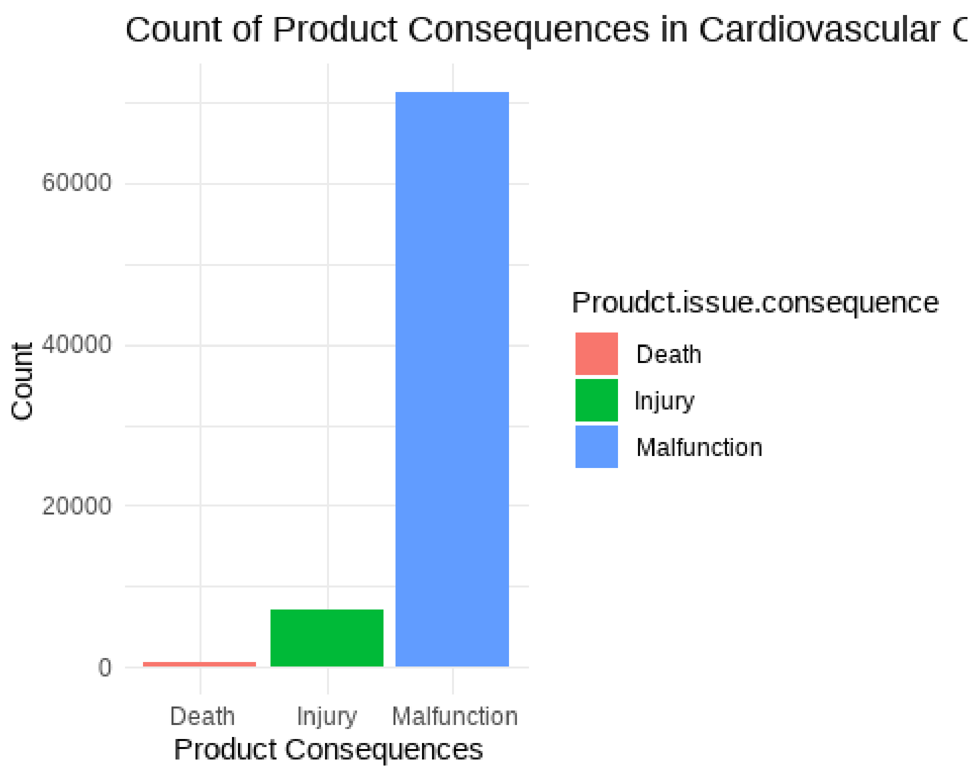
sub-seting the data based on the product field description to better understand the trend in the data

```
# Subsetting for Cardiovascular
df_cardiovascular <- df1 %>%
  filter(product.field_description == "Cardiovascular")

df_clinical_chemistry <- df1 %>%
  filter(product.field_description == "Clinical Chemistry")

df_unknown <- df1 %>%
  filter(product.field_description == "Unknown")

df_general_hospital <- df1 %>%
  filter(product.field_description == "General Hospital")
```

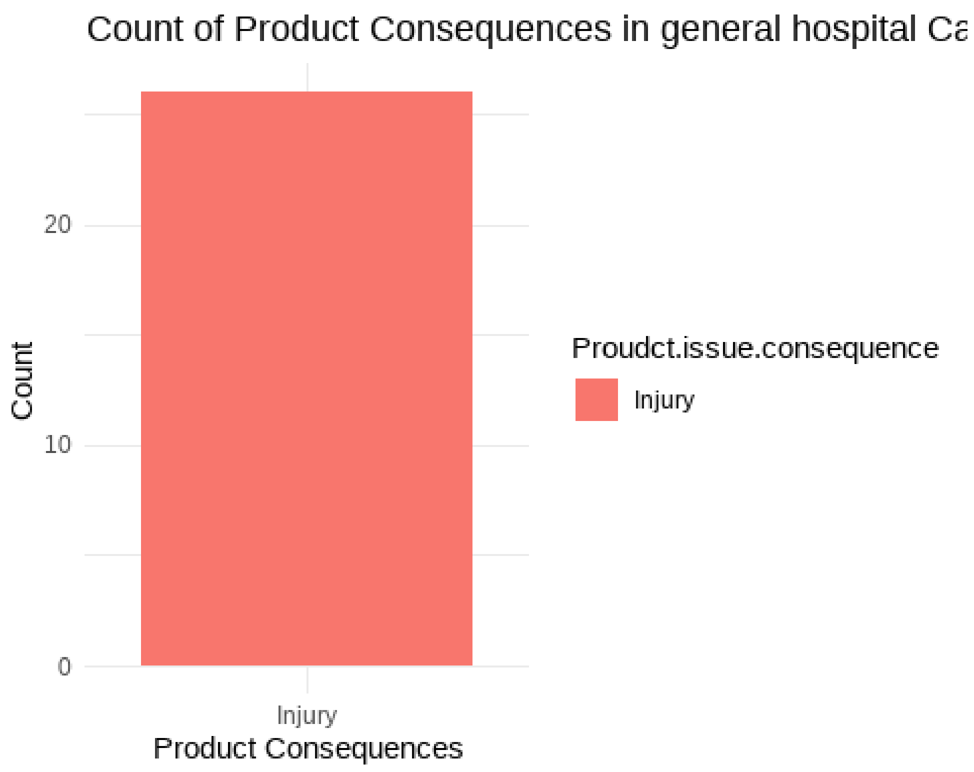
- In the Cardiovascular category, there is a significant prevalence of malfunctions compared to non-malfunctions.
- The General Hospital category shows only non-malfunctions with a relatively low count.
- For the Clinical Chemistry category, both malfunctions and non-malfunctions are present, with malfunctions occurring more frequently.
- The Unknown category has a large number of malfunctions and a substantial number of non-malfunctions.

These charts provide a clear visual comparison of the frequency of product issue consequences, highlighting the dominant issue type in each category. It's notable that "Malfunction" appears to be the more common type of consequence across most categories, except for the General Hospital category where only non-malfunctions are reported.

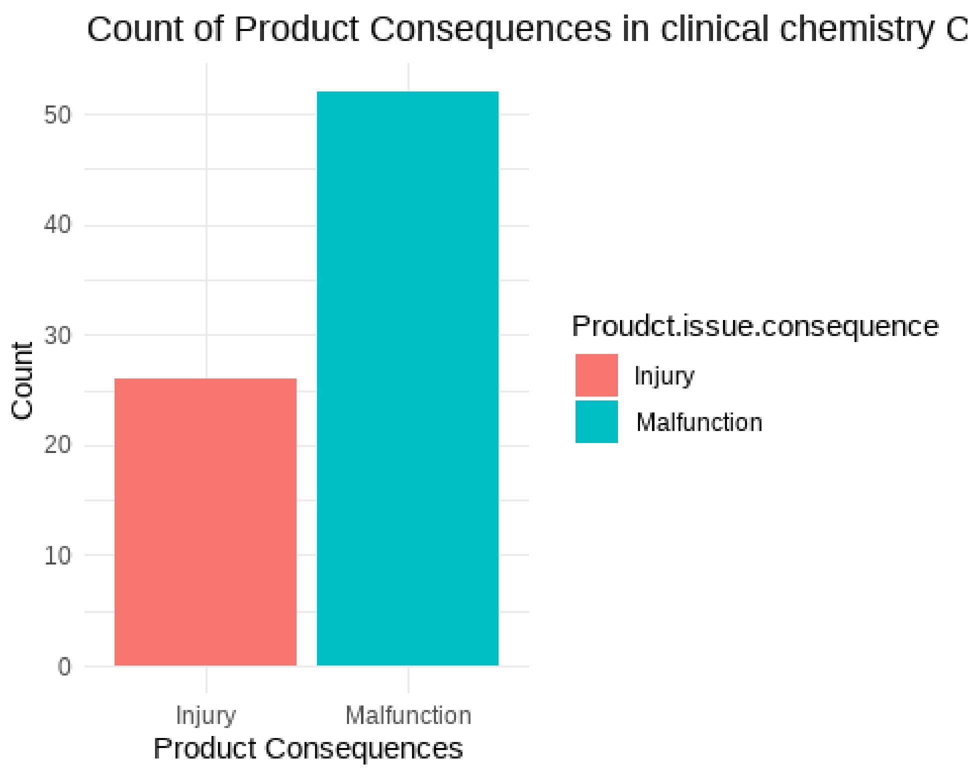
```r
ggplot(df_cardiovascular, aes(x = Proudct.issue.consequence)) +
  geom_bar(aes(fill = Proudct.issue.consequence), position = "dodge") +
  theme_minimal() +
  labs(title = "Count of Product Consequences in Cardiovascular Category",
       x = "Product Consequences",
       y = "Count")
```



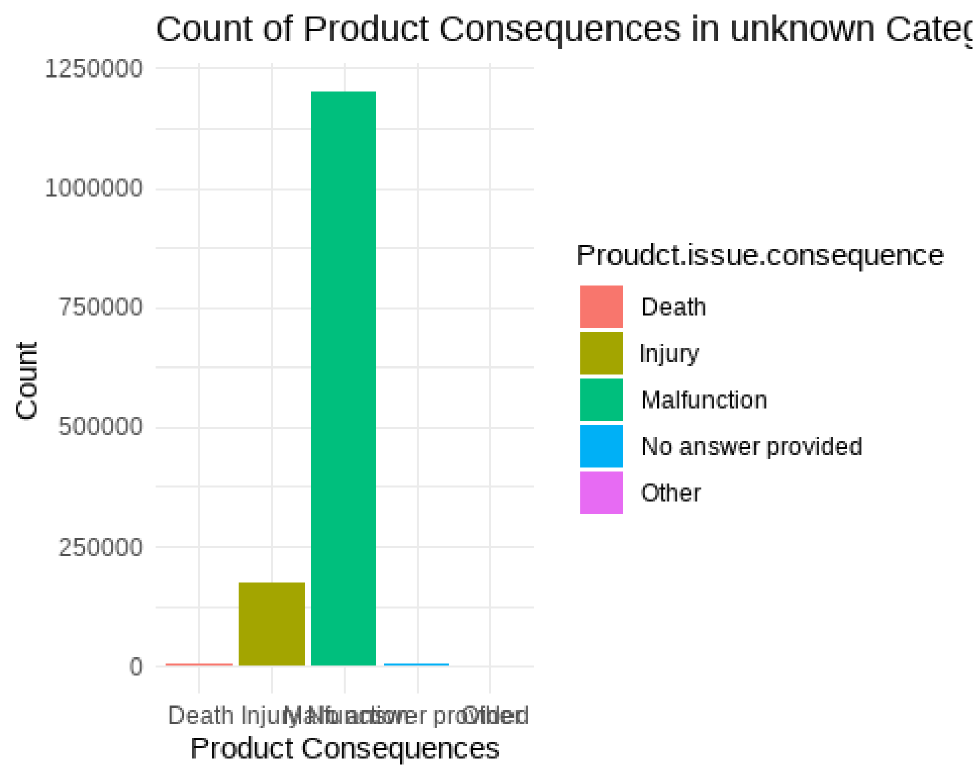
```r
ggplot(df_general_hospital, aes(x = Proudct.issue.consequence)) +
  geom_bar(aes(fill = Proudct.issue.consequence), position = "dodge") +
  theme_minimal() +
  labs(title = "Count of Product Consequences in general hospital Category",
       x = "Product Consequences",
       y = "Count")
```



```
ggplot(df_clinical_chemistry, aes(x = Proudct.issue.consequence)) +
  geom_bar(aes(fill = Proudct.issue.consequence), position = "dodge") +
  theme_minimal() +
  labs(title = "Count of Product Consequences in clinical chemistry
Category",
       x = "Product Consequences",
       y = "Count")
```



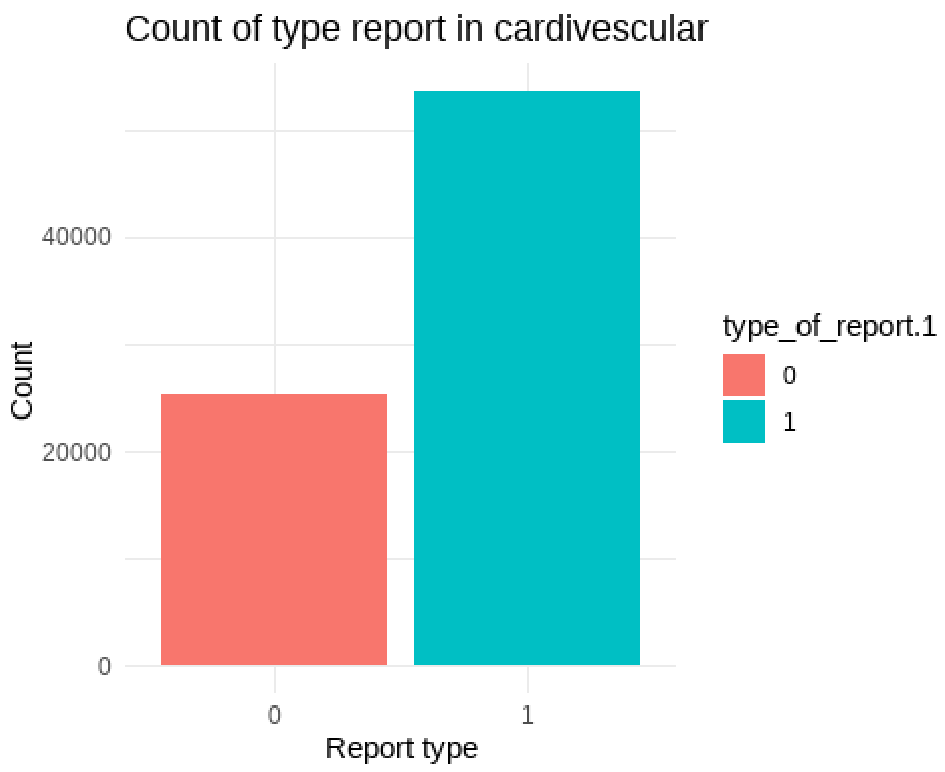
```
ggplot(df_unknown, aes(x = Proudct.issue.consequence)) +
  geom_bar(aes(fill = Proudct.issue.consequence), position = "dodge") +
  theme_minimal() +
  labs(title = "Count of Product Consequences in unknown Category",
       x = "Product Consequences",
       y = "Count")
```



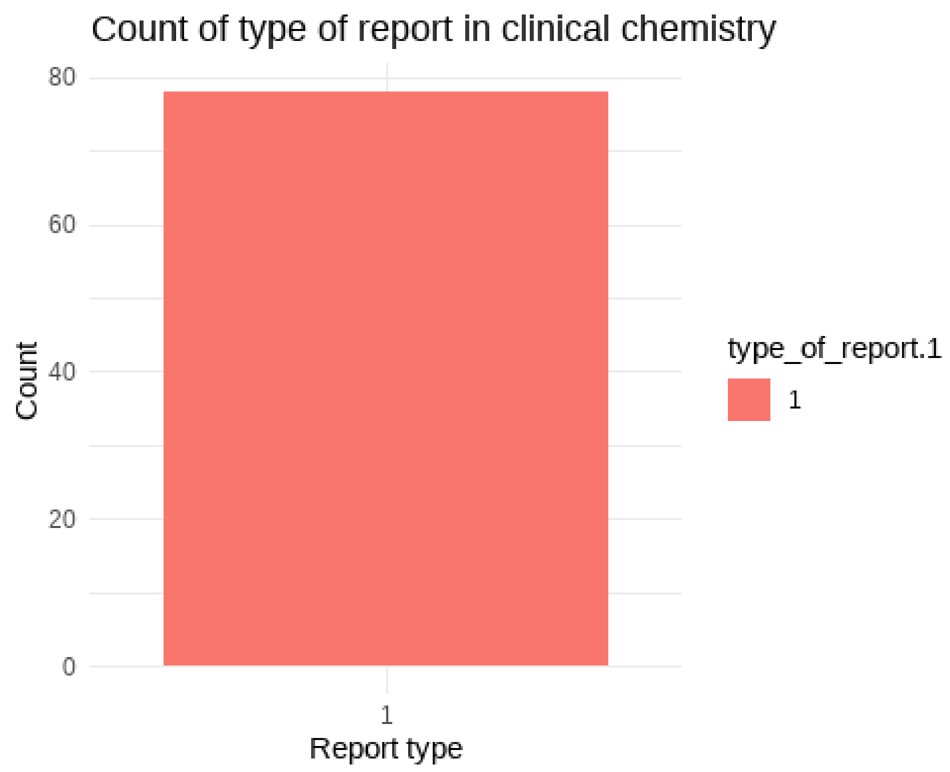
* In the cardiovascular category, there are significantly more reports of type “1” than type “0”. * The clinical chemistry category shows only reports of type “1”. * The general hospital category similarly shows reports exclusively of type “1”. * In the unknown category, reports of type “1” are also more common than type “0”, but both are present.

The bar colors represent the two report types, with one color for type “0” and another for type “1”. The exact nature of these report types isn’t specified, but it’s clear that type “1” is more prevalent in these categories, with the exception of cardiovascular, where type “0” also appears in a substantial count.

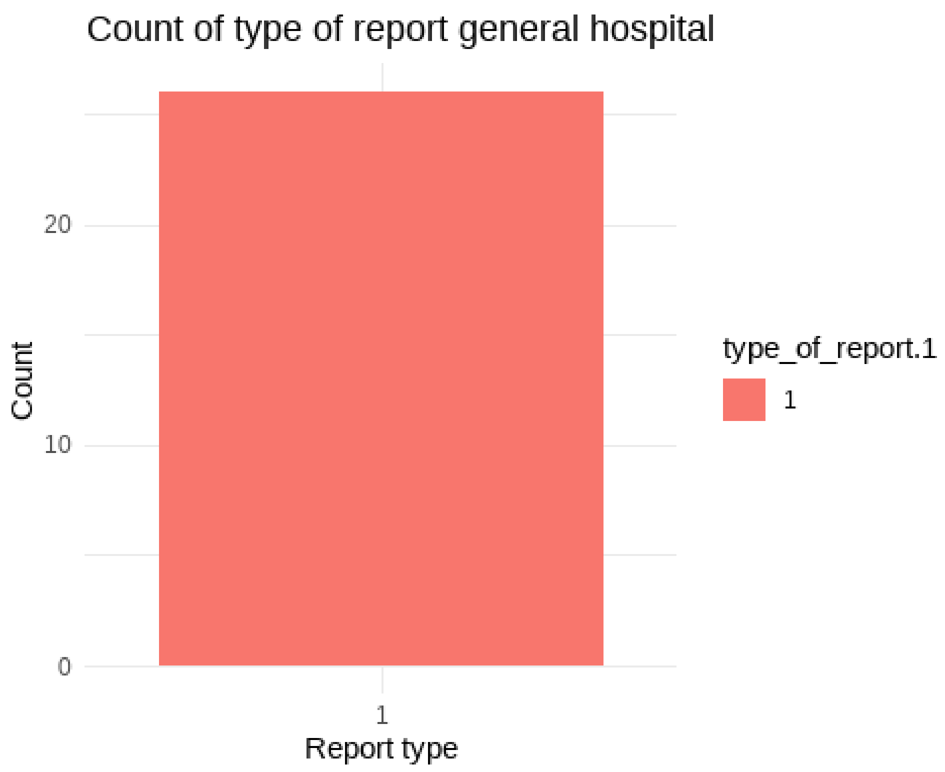
```
ggplot(df_cardiovascular, aes(x = type_of_report.1)) +
  geom_bar(aes(fill = type_of_report.1), position = "dodge") +
  theme_minimal() +
  labs(title = "Count of type report in cardivescular",
       x = "Report type",
       y = "Count")
```



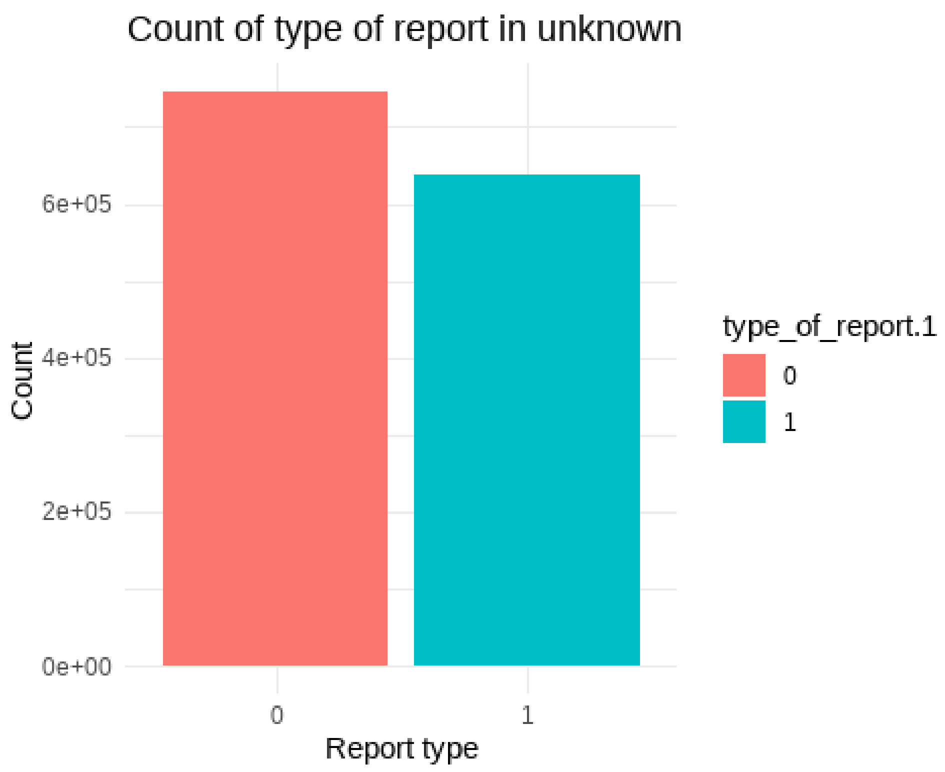
```
ggplot(df_clinical_chemistry, aes(x = type_of_report.1)) +
  geom_bar(aes(fill = type_of_report.1), position = "dodge") +
  theme_minimal() +
  labs(title = "Count of type of report in clinical chemistry",
       x = "Report type",
       y = "Count")
```



```r
ggplot(df_general_hospital, aes(x = type_of_report.1)) +
  geom_bar(aes(fill = type_of_report.1), position = "dodge") +
  theme_minimal() +
  labs(title = "Count of type of report general hospital",
       x = "Report type",
       y = "Count")
```



```
ggplot(df_unknown, aes(x = type_of_report.1)) +
  geom_bar(aes(fill = type_of_report.1), position = "dodge") +
  theme_minimal() +
  labs(title = "Count of type of report in unknown",
       x = "Report type",
       y = "Count")
```


Count of type of report in unknown
Count
6e+05
4e+05
2e+05
0e+00
0
1
Report type
type_of_report.1
0
1

Feature engineearing - B1

In our feature engineering process for Task B1, we aim to simplify the analysis by reclassifying the ‘Proudct.issue.consequence’ variable into two categories: ‘Malfunction’ and ‘Non-Malfunction’. This binary classification will streamline our dataset, making it more conducive to predictive modeling.

```
df1$Proudct.issue.consequence <- ifelse(df1$Proudct.issue.consequence !=
"Malfunction", "Non-Malfunction", "Malfunction")

df1$Proudct.issue.consequence <- as.factor(df1$Proudct.issue.consequence)


table(df1$Proudct.issue.consequence)

##
##     Malfunction Non-Malfunction
##         1272089          190722
```

As we transition to addressing class imbalance, it’s crucial to note the significant discrepancy in our target variable’s classes. Our dataset exhibits a pronounced imbalance, with ‘Malfunction’ occurrences vastly outnumbering ‘Non-Malfunction’ ones. This disparity can skew our model’s accuracy, necessitating a strategy to create a more balanced dataset.

To mitigate the impact of class imbalance on our model’s performance, we employ an undersampling strategy. By randomly reducing the ‘Malfunction’ class to match the ‘Non-Malfunction’ class size, we aim to achieve a balanced representation. This approach enhances the model’s ability to learn from both classes equally, improving its generalization capability.

```
# Stratified sampling of the majority class
set.seed(123) # for reproducibility

# Determine the size of the minority class
minority_size <- table(df1$Proudct.issue.consequence)["Non-Malfunction"]

# Create indices for each class
indices_majority <- which(df1$Proudct.issue.consequence == "Malfunction")
indices_minority <- which(df1$Proudct.issue.consequence != "Malfunction")

# Stratified random sampling from the majority class
set.seed(123) # Ensure reproducibility
indices_majority_sample <- sample(indices_majority, minority_size)

# Combine the indices from both classes
final_indices <- c(indices_majority_sample, indices_minority)

# Create the balanced dataset
df1_balanced_under <- df1[final_indices, ]
```

```r
# Verify the balance
table(df1_balanced_under$Proudct.issue.consequence)

##
##     Malfunction Non-Malfunction
##          190722          190722
```

Converting Response Variable into Binary

Finally, to prepare our dataset for the Lasso regression model, we convert the 'Proudct.issue.consequence' variable into a binary format. This step involves coding 'Malfunction' as 1 and 'Non-Malfunction' as 0. By doing so, we simplify the model's output, focusing on a clear binary prediction: whether an instance is a 'Malfunction' or not. This binary response variable is crucial for applying logistic regression techniques effectively.

```r
df1_balanced <- df1_balanced_under %>%
  mutate(Proudct.issue.consequence = as.integer(Proudct.issue.consequence ==
"Malfunction"))


table(df1_balanced$Proudct.issue.consequence)

##
##      0      1
## 190722 190722
```

In preparation for Lasso regression, we refine our dataset by excluding variables that may not contribute to our predictive model. This step ensures a focused analysis, reducing complexity and enhancing computational efficiency.

```r
#library(dplyr)
df2 <- df1_balanced %>%
  dplyr::select(-ID_non_uniq, -date_event)
```

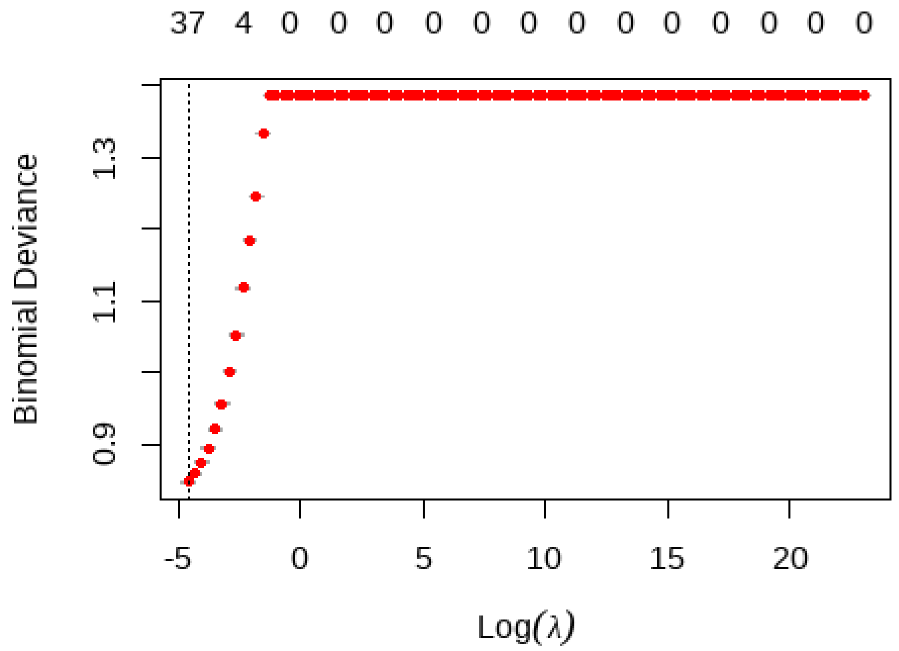
LASSO Base model

Given the potential high dimensionality of our data after creating dummy variables, we opt for a sparse matrix representation. This choice is strategic for managing memory and computational resources more efficiently. Sparse matrices store only non-zero elements, drastically reducing the amount of memory needed compared to a dense matrix format. This efficiency is particularly beneficial when applying Lasso regression, as it involves iterative processes over potentially large sets of predictors. By utilizing a sparse matrix, we maintain the scalability and speed of our model fitting process.

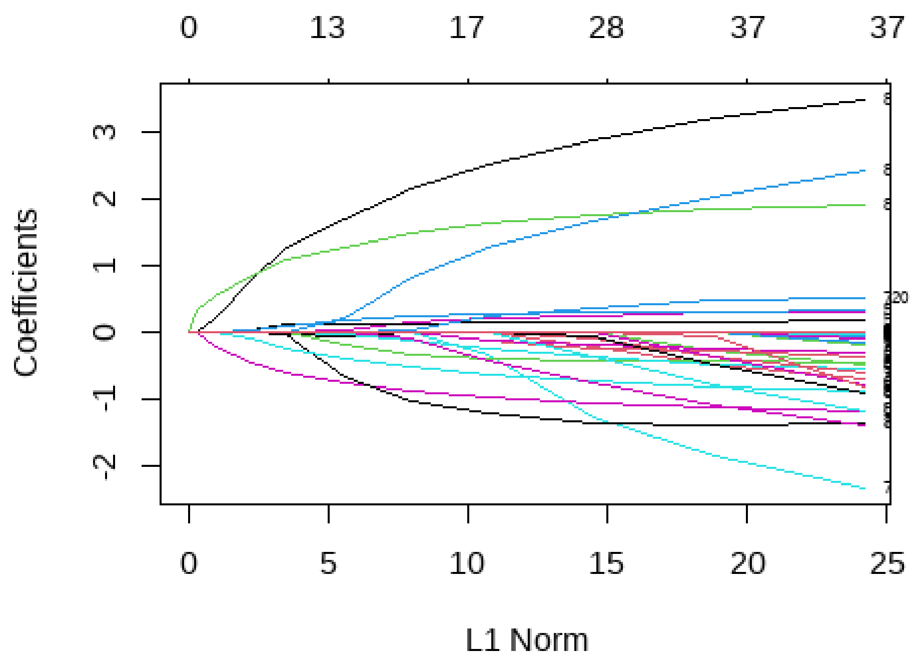
In R, you can easily scale and normalize your data before running a Lasso regression. If you are using the glmnet package, which is commonly used for Lasso (and Elastic Net models), we can actually let the function handle scaling for you. By default, glmnet normalizes the features (independent variables) before fitting the model and then returns the coefficients on the original scale.

As we proceed with Lasso regression, the aim is to leverage its ability to perform both variable selection and regularization. This approach not only helps in identifying the most predictive features but also in avoiding overfitting by penalizing the size of the coefficients. The regularization strength, dictated by lambda (λ), is optimized through cross-validation to find the balance that minimizes prediction error

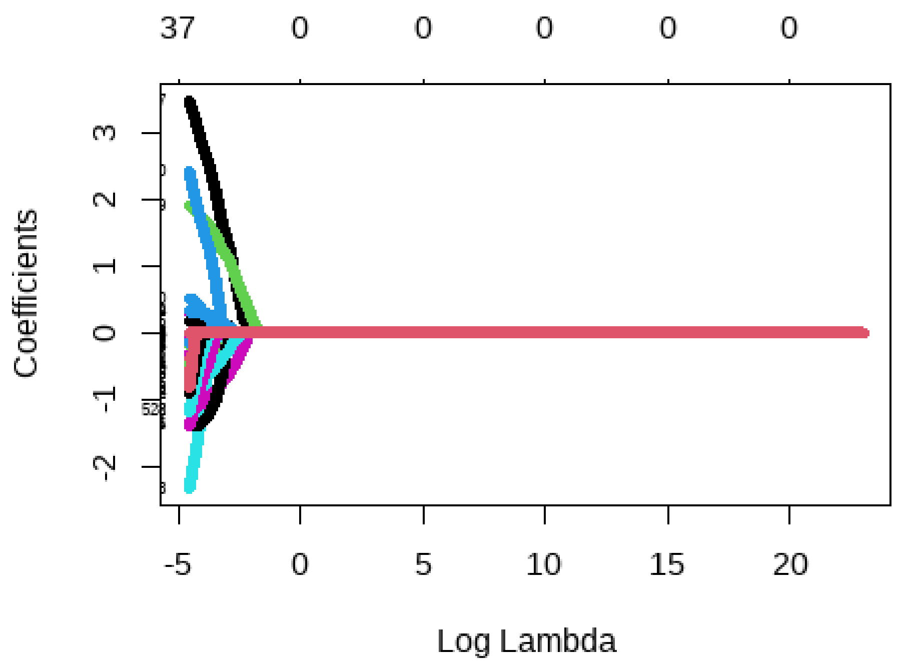
```r
#library(glmnet)
#library(Matrix)

# Dropping non needed column
df3 <- df2[-1]

# Creating dummy variables of predictor variables using a sparse matrix
X <- sparse.model.matrix(~ . - Proudct.issue.consequence, data = df3)

# Separating the response variable
y <- df3$Proudct.issue.consequence

# Splitting train and test data
set.seed(40418446)
train_indices <- sample(1:nrow(X), size = nrow(X) * 0.5)
test_indices <- setdiff(1:nrow(X), train_indices)

# Extracting the response variable for the test set
y_test <- y[test_indices]

# Creating a lambda sequence
grid <- 10^seq(10, -2, length = 100)

# Fit Lasso model with cross-validation
cv_model_main <- cv.glmnet(X[train_indices, ], y[train_indices], family =
"binomial", alpha = 1, lambda = grid, type.measure = "deviance", nfolds = 10)
plot(cv_model_main)
```



```
# Select the best lambda value
best_lambda_main <- cv_model_main$lambda.min

# Fitting Lasso model on the training set
lasso.mod_main <- glmnet(X[train_indices, ], y[train_indices], family =
"binomial", type.measure = "deviance", alpha = 1, lambda = grid)
plot(lasso.mod_main, label = TRUE)

## Warning in regularize.values(x, y, ties, missing(ties), na.rm = na.rm):
## collapsing to unique 'x' values
```



```r
plot(lasso.mod_main, xvar = "lambda", label = TRUE, lwd = 6)
```



```r
# Prediction using Lasso
lasso.pred_main <- predict(lasso.mod_main, newx = X[test_indices, ], s =
best_lambda_main, type = "response")
pred_class_main <- ifelse(lasso.pred_main >= 0.5, 1, 0)
accuracy_main <- mean(pred_class_main == y_test)
print(accuracy_main)
```

```
## [1] 0.8001437
```

```r
# Running Lasso on the whole dataset
out_main <- glmnet(X, y, alpha = 1, family = "binomial", type.measure =
"deviance", lambda = grid)
```

The base Lasso model, with a prediction accuracy of approximately 80%, suggests a strong ability to correctly classify cases as 'Malfunction' or 'Non-Malfunction'. Here are some insights based on the results and plots provided:

- **Model Fit and Selection**: The model has identified a set of predictors that contribute to the outcome variable 'Proudct.issue.consequence'. With 37 predictors remaining non-zero, the Lasso regression has effectively performed feature selection, potentially improving model interpretability by eliminating irrelevant features.

- **Optimal Lambda**: The optimal lambda value chosen through cross-validation minimizes the binomial deviance, striking a balance between model complexity and prediction accuracy.

- **Coefficient Paths**: The coefficient path plot shows how the absolute values of the coefficients shrink towards zero as lambda increases. Only a subset of features remains with non-zero coefficients, indicating their importance in predicting the outcome.

- **Coefficient Shrinkage and Selection**: The coefficient plot against the L1 norm demonstrates the shrinkage effect of Lasso. The model starts with all coefficients at their non-regularized values when lambda is zero and progressively shrinks some coefficients to zero as lambda increases. This regularization process helps in avoiding overfitting and in selecting features that have the most predictive power.

- **Accuracy**: An accuracy rate of 80% in a binary classification setting is quite high and suggests that the model is performing well.

Post-model fitting, we extract and examine the coefficients, filtering to retain only those variables with non-zero coefficients. This outcome highlights the features that the Lasso regression deems most influential in predicting the target variable. By focusing on these predictors, we gain insights into the underlying relationships within our data, informing further analysis and decision-making.

```r
# Extracting coefficients of the Lasso model
lasso_coef_main <- predict(out_main, type = "coefficients", s =
best_lambda_main)
```

```r
# Convert to a regular dense format (excluding the intercept)
lasso_coef_dense_main <- as.numeric(lasso_coef_main[-1, ])

# Get the names of the coefficients
coef_names_main <- rownames(lasso_coef_main)

# Create a data frame with variable names and their non-zero coefficients
lc_df <- data.frame(
  variable = coef_names_main[-1],
  coefficient = lasso_coef_dense_main
)

# Filter out zero coefficients
lc_df <- lc_df[lc_df$coefficient != 0, ]

# Rank the coefficients by their absolute values
lc_df$abs_coefficient <- abs(lc_df$coefficient)
lc_df <- lc_df[order(-lc_df$abs_coefficient), ]

# Select only the non-zero coefficients and their names for the list
important_features <- lc_df$variable

# Print the ranked list of important features
print(important_features)
```

```
##  [1] "product_issue_type597"
##  [2] "product_issue_type600"
##  [3] "manufacturer_contact_address_16601"
##  [4] "product_issue_type599"
##  [5] "product_issue_type596"
##  [6] "product.product_report_product_codeDXY"
##  [7] "product_issue_type816"
##  [8] "type_of_report.11"
##  [9] "product_issue_type955"
## [10] "product_issue_type839"
## [11] "product_issue_type906"
## [12] "product_issue_type413"
## [13] "product.generic_name44310"
## [14] "product_issue_type212"
## [15] "source_type12"
## [16] "product.manufacturer_country126"
## [17] "reporter_job_code52"
## [18] "product.generic_name73622"
## [19] "reporter_job_code42"
## [20] "product.brand_name215593"
## [21] "source_type10"
## [22] "source_type17"
## [23] "reporter_job_code32"
## [24] "product_issue_type222"
## [25] "product.manufacturer_country49"
## [26] "product_issue_type350"
## [27] "product.generic_name101028"
## [28] "product.brand_name215539"
## [29] "product.manufacturer_name13738"
## [30] "last_year_root_cause_description_most_freq"
## [31] "product_issue_type598"
## [32] "product.issue.type"
## [33] "last_two_years_decision_date_max_changes_in_product"
## [34] "last_year_classification1_num_uniq"
## [35] "last_two_years_classification1_num_uniq"
## [36] "last_two_years_decision_date_average_changes_in_product"
## [37] "product.manufacturer_city2024"
```

Dataset subseting based on "product.brand.name"

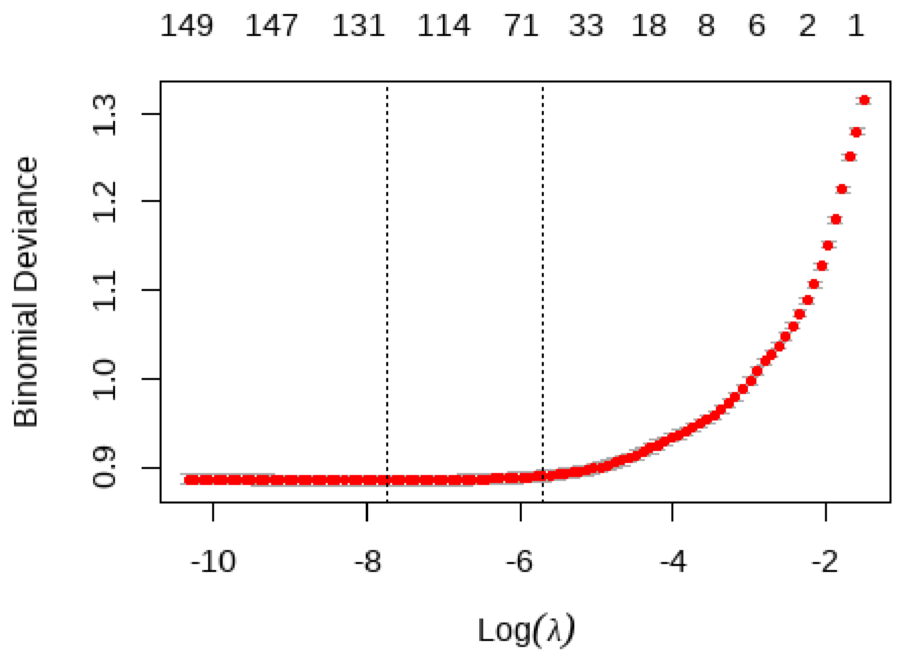
To evaluate the model on different segments to know the predictability and accuracy of the model the data was divided into two dataframes: subset_A which includes only the rows with product.brand_name equal to "215795", and subset_B which includes all rows with product.brand_name not equal to "215795".

```r
# Subset where product.brand_name is '215795'
subset_A <- df2[df2$product.brand_name == '215795', ]

# Subset where product.brand_name is not '215795'
subset_B <- df2[df2$product.brand_name != '215795', ]

set.seed(123) # Set a seed for reproducibility
n <- 50000 # Number of observations to sample

sampled_dfA <- subset_A %>%
  sample_n(size = n)

set.seed(123) # Set a seed for reproducibility
n <- 50000 # Number of observations to sample

sampled_dfB <- subset_B %>%
  sample_n(size = n)
```

Lasso model with subset A

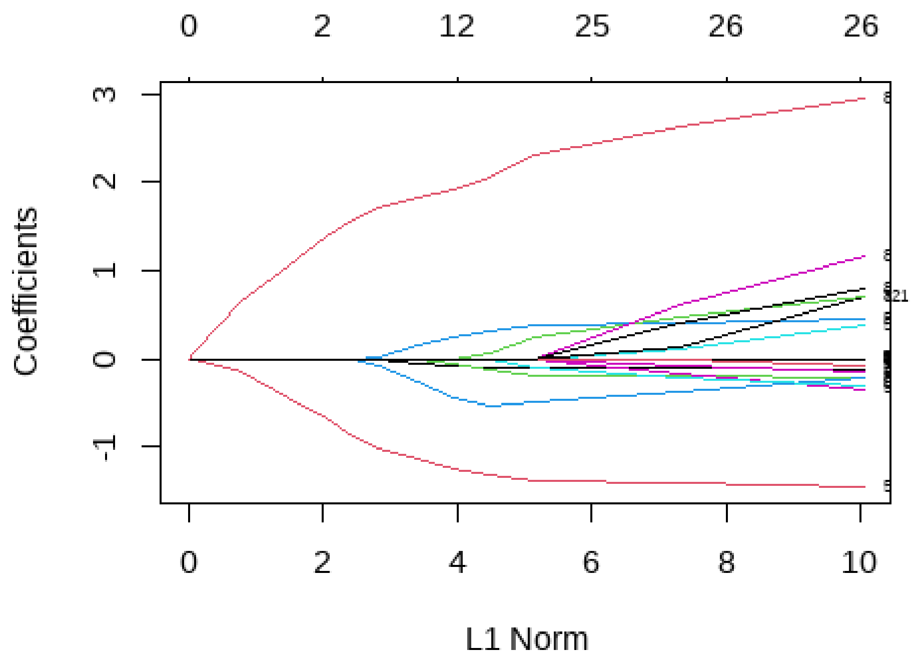
```r
# creating dummy variables of predictor variables
x = sparse.model.matrix(Proudct.issue.consequence~.,sampled_dfA)[,-1]

# separating response variable
y= sampled_dfA$Proudct.issue.consequence



# splitting train and test data
set.seed(40418446)
```

```
train.l = sample(1:nrow(x), nrow(x)/2)
test.l = (-train.l)
y.test.l = y[test.l]


# creating a lambda sequence
grid = 10^seq(10, -2, length = 100)

# Fit Lasso model with cross-validation to find best lambda
cv_modelA <- cv.glmnet(x[train.l,], y[train.l], family="binomial", alpha=1,
type.measure = "deviance", nfolds =10)

# Plot the cross-validation error as a function of lambda
plot(cv_modelA)
```

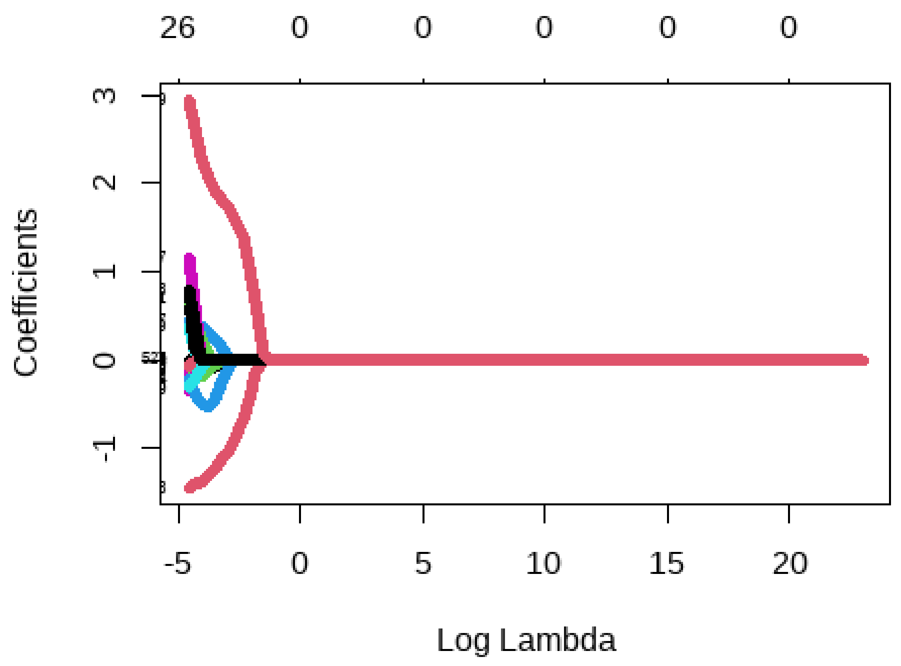


```
# Select the best lambda value
best_lambdaA <- cv_modelA$lambda.min
best_lambdaA

## [1] 0.0004380732

# fitting lasso model on train set
lasso.mod_A=glmnet(x[train.l,], y[train.l],family="binomial",type.measure =
"deviance", alpha=1, lambda=grid)
plot(lasso.mod_A, label=TRUE)
```

```
## Warning in regularize.values(x, y, ties, missing(ties), na.rm = na.rm):
## collapsing to unique 'x' values
```



```
plot(lasso.mod_A, xvar="lambda", label=TRUE, lwd=6)
```



```r
# prediction using lasso
lasso.pred <- predict(lasso.mod_A,family="binomial", s = best_lambdaA, newx =
x[test.l,])
pred_class <- ifelse(lasso.pred >= 0.5, 1, 0)
accuracy <- mean(pred_class == y[test.l])
accuracy

## [1] 0.75368

# running lasso on whole data set
lasso_coef_A <- predict(lasso.mod_A, type = "coefficients", s = best_lambdaA)

# Convert to a regular dense format (excluding the intercept if present)
lasso_coef_dense_A <- as.numeric(lasso_coef_A[-1, ])  # Skip this if you want
to include the intercept

# Get the names of the coefficients
coef_names_A <- rownames(lasso_coef_A)

# Create a data frame with variable names and their non-zero coefficients
lc_dfA <- data.frame(
  variable = coef_names_A[-1],  # Exclude the intercept's name if present
  coefficient = lasso_coef_dense_A
)

# Filter out zero coefficients
lc_dfA <- lc_dfA[lc_dfA$coefficient != 0, ]
```

```
# Rank the coefficients by their absolute values
lc_dfA$abs_coefficient <- abs(lc_dfA$coefficient)
lc_dfA <- lc_dfA[order(-lc_dfA$abs_coefficient), ]

# Select only the non-zero coefficients and their names for the list
important_features_A <- lc_dfA$variable

# Print the ranked list of important features
print(important_features_A)
```

```
##  [1] "product_issue_type599"
##  [2] "type_of_report.11"
##  [3] "product_issue_type597"
##  [4] "product_issue_type598"
##  [5] "product_issue_type399"
##  [6] "product_issue_type168"
##  [7] "source_type6"
##  [8] "source_type10"
##  [9] "source_type12"
## [10] "product_issue_type596"
## [11] "product_issue_type413"
## [12] "reporter_job_code52"
## [13] "product.generic_name101028"
## [14] "last_two_years_decision_date_max_changes_in_product"
## [15] "product_issue_type212"
## [16] "last_year_reason_for_legal_announcement_num_uniq"
## [17] "last_two_years_reason_for_legal_announcement_num_uniq"
## [18] "last_two_years_decision_date_average_changes_in_product"
## [19] "last_four_years_reason_for_legal_announcement_num_uniq"
## [20] "product.issue.type"
## [21] "last_year_classification1_num_uniq"
## [22] "last_two_years_classification1_num_uniq"
## [23] "last_four_years_classification1_num_uniq"
## [24] "last_two_years_reason_for_legal_announcement_most_freq"
## [25] "last_year_reason_for_legal_announcement_most_freq"
## [26] "last_four_years_reason_for_legal_announcement_most_freq"
```

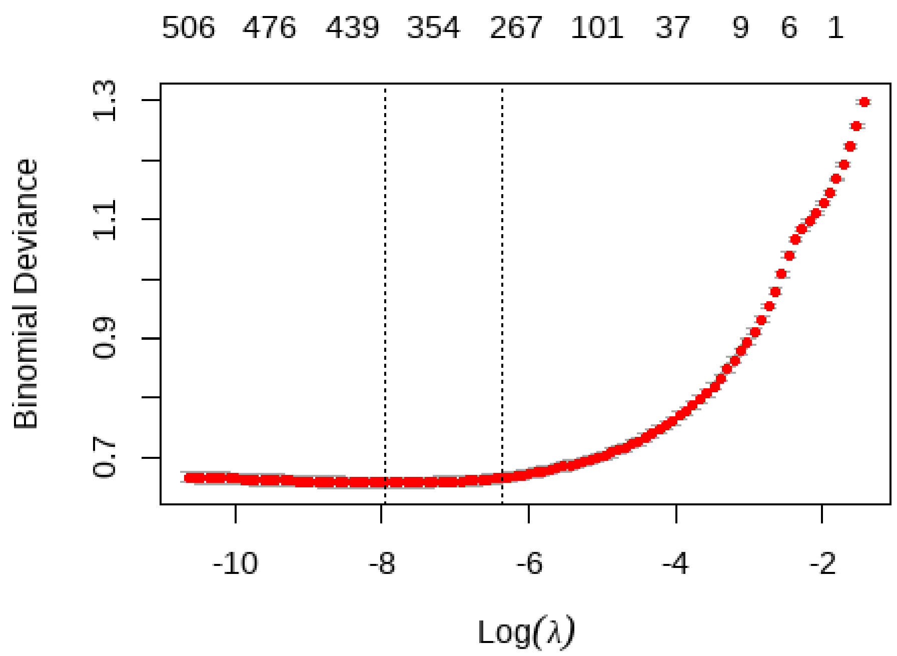
key insights:

- **Model Summary**: The cross-validated Lasso model, cv_model1A, determined an optimal lambda (λ) through binomial deviance minimization. With 175 features starting non-zero and the best lambda yielding 94 non-zero features, the model has effectively performed feature selection to reduce complexity.
- **Coefficient Path Plot**: The coefficients plot against the L1 norm shows how coefficients are penalized. As lambda increases, more coefficients are shrunk towards zero, leaving only the most influential features in the model.

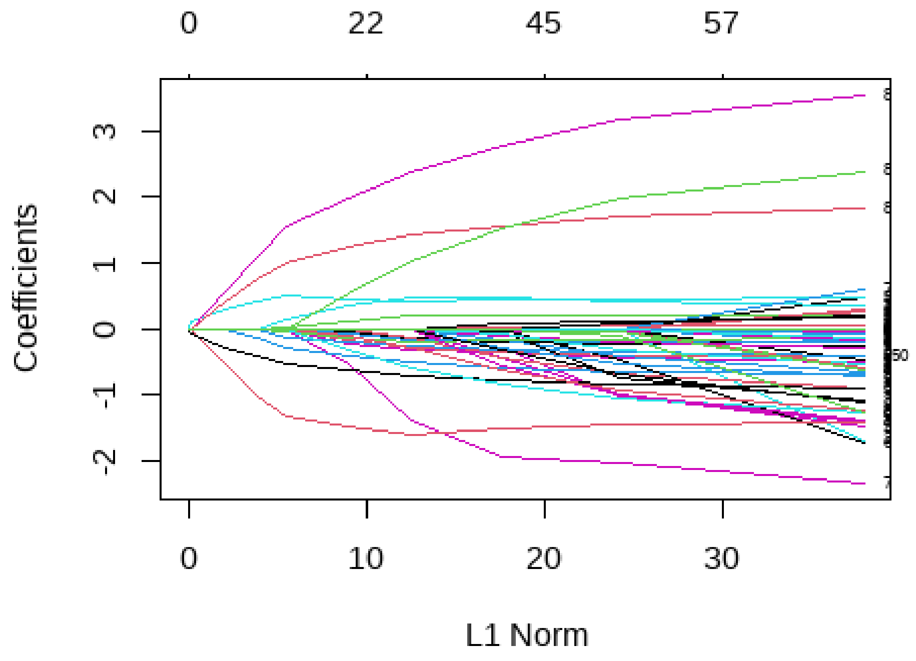
- **Lambda vs. Binomial Deviance Plot**: The curve shows how the model deviance changes as lambda increases. A lower deviance at a specific lambda indicates a better-fitting model.
- **Accuracy of Model**: An accuracy of about 75% is fairly strong, but it is slightly lower than your base model. This could be due to over-penalization, or it might indicate the subset model is more generalized and possibly less overfit than the base model.
- **Comparison with Base Model**: Comparing to your base model accuracy of 80%, the subset model has a modest reduction in predictive performance. This might be due to the model capturing less variance or being more robust to overfitting, which is a common trade-off in model complexity and generalizability.

Lasso model with subset B

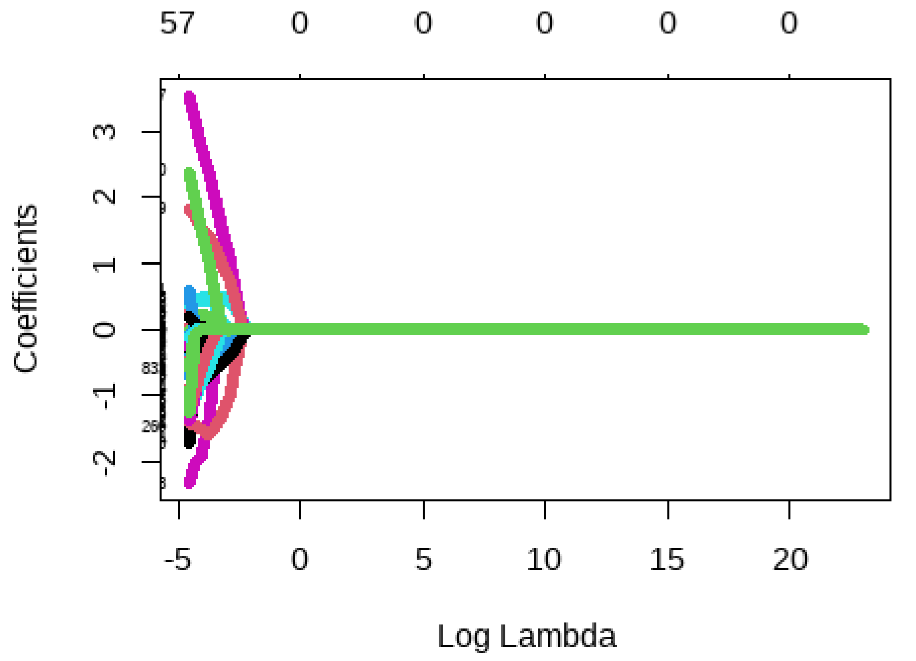
```
# creating dummy variables of predictor variables
x = sparse.model.matrix(Proudct.issue.consequence~.,sampled_dfB)[,-1]

# separating response variable
y = sampled_dfB$Proudct.issue.consequence


# splitting train and test data
set.seed(40418446)
train.l = sample(1:nrow(x), nrow(x)/2)
test.l = (-train.l)
y.test.l = y[test.l]


# creating a lambda sequence
grid = 10^seq(10, -2, length = 100)

# Fit Lasso model with cross-validation to find best lambda
set.seed(4200)
cv_modelB <- cv.glmnet(x[train.l,], y[train.l], family="binomial", alpha=1,
type.measure = "deviance", nfolds =10)

# Plot the cross-validation error as a function of lambda
plot(cv_modelB)
```



```
# Select the best lambda value
best_lambdaB <- cv_modelB$lambda.min
best_lambdaB

## [1] 0.0003545109

# fitting lasso model on train set
lasso.mod_B=glmnet(x[train.l,], y[train.l],family="binomial",type.measure =
"deviance", alpha=1, lambda=grid)
plot(lasso.mod_B, label=TRUE)

## Warning in regularize.values(x, y, ties, missing(ties), na.rm = na.rm):
## collapsing to unique 'x' values
```



```r
plot(lasso.mod_B, xvar="lambda", label=TRUE, lwd=6)
```



```
# prediction using lasso
lasso.pred_B <- predict(lasso.mod_B,family="binomial", s = best_lambdaB, newx
= x[test.l,])
pred_class_B <- ifelse(lasso.pred_B >= 0.5, 1, 0)
accuracy_B <- mean(pred_class_B == y[test.l])
accuracy_B

## [1] 0.82648

# running lasso on whole data set
out_B = glmnet(x, y, alpha=1, family="binomial", type.measure = "deviance",
lambda = grid)

# extracting coeficients of lasso model
lasso_coef_B <- predict(out_B, type = "coefficients", s = best_lambdaB)

# Convert to a regular dense format (excluding the intercept if present)
lasso_coef_dense_B <- as.numeric(lasso_coef_B[-1, ])  # Skip this if you want
to include the intercept

# Get the names of the coefficients
coef_names_B <- rownames(lasso_coef_B)

# Create a data frame with variable names and their non-zero coefficients
lc_dfB <- data.frame(
  variable = coef_names_B[-1],  # Exclude the intercept's name if present
  coefficient = lasso_coef_dense_B
)

# Filter out zero coefficients
lc_dfB <- lc_dfB[lc_dfB$coefficient != 0, ]

# View the resulting data frame
print(lc_dfB)

##                                                 variable   coefficient
## 6                    last_year_classification1_num_uniq  1.993952e-05
## 15           last_year_root_cause_description_most_freq -1.706306e-02
## 26                last_two_years_classification1_num_uniq  3.637796e-07
## 33   last_two_years_legal_announcementing_firm_most_freq  4.585458e-04
## 46               last_four_years_classification1_num_uniq  7.759313e-08
## 78                    manufacturer_contact_address_16601 -2.611217e+00
## 95                               product.brand_name19264 -8.811029e-02
## 116                              product.brand_name43807  1.961628e-01
## 130                              product.brand_name76920 -1.473293e-01
## 224                             product.brand_name180345 -1.423338e+00
## 260                             product.brand_name215539 -1.155629e+00
## 265                             product.brand_name215546  6.157904e-01
## 304                             product.brand_name215593  9.828970e-02
## 374                             product.brand_name215851  3.112026e-01
## 482                        product.generic_name18090 -1.178741e-01
## 487                        product.generic_name44310 -1.027044e+00
## 492                        product.generic_name46965  1.778671e-01
## 514                        product.generic_name73622 -6.785448e-01
## 527                               product.issue.type  3.059777e-03
## 528                                type_of_report.11 -7.252679e-01
## 535                               reporter_job_code32  4.692154e-01
## 541                               reporter_job_code42 -5.467447e-01
## 544                               reporter_job_code52 -3.681537e-01
## 549                                     source_type10  1.877291e-01
## 550                                     source_type12 -3.591423e-01
## 553                                     source_type17 -3.440410e-01
## 578                  product.manufacturer_name13738 -6.216783e-02
## 610                  product.manufacturer_name19392 -1.185243e+00
## 614                  product.manufacturer_name19402  6.118795e-01
## 665                   product.manufacturer_city3154 -2.912547e-01
## 714                  product.manufacturer_country49 -3.980038e-01
## 716                  product.manufacturer_country79 -1.371758e-03
## 720                 product.manufacturer_country126  4.792845e-01
## 727          product.product_report_product_codeDRF -1.141992e-01
## 728          product.product_report_product_codeDTB -7.917029e-01
## 731          product.product_report_product_codeDXY -1.250468e+00
## 742          product.product_report_product_codeOZO -2.474440e-01
## 785                           product_issue_type212 -7.162582e-01
## 786                           product_issue_type222 -3.734372e-01
## 793                           product_issue_type256 -1.615887e-02
## 824                           product_issue_type413 -9.583544e-01
## 838                           product_issue_type547 -5.119074e-01
## 846                           product_issue_type596 -1.154160e+00
## 847                           product_issue_type597  3.752607e+00
## 848                           product_issue_type598  2.951029e-01
## 849                           product_issue_type599  1.901245e+00
## 850                           product_issue_type600  2.578754e+00
## 865                           product_issue_type708 -5.986813e-01
## 882                           product_issue_type816 -1.175676e+00
## 885                           product_issue_type839 -7.059561e-01
## 892                           product_issue_type906 -7.942942e-01
## 898                           product_issue_type964 -6.704716e-01
```

Key Insights:

- **Model Performance**: The third Lasso model outperforms the base model in terms of accuracy, indicating better predictive performance on the subset B of the data.
- **Optimal Lambda Selection**: Similar to previous models, the optimal lambda value is selected to minimize binomial deviance. This model strikes a balance between capturing the underlying data structure and preventing overfitting, as indicated by its lower deviance at the chosen lambda compared to the base model.

- **Coefficient Shrinking**: The coefficients path plot demonstrates the effectiveness of Lasso in shrinking less relevant predictors towards zero. The plot against the L1 norm confirms that the model maintains a robust selection of features that contribute to predicting the outcome.
- **Model Complexity**: The model starts with 406 non-zero features at the smallest lambda and simplifies to 286 non-zero features at the optimal lambda, indicating a reduction in model complexity while still maintaining strong predictive power.
- **Comparative Insight**: When compared to the base model, the improvement in accuracy suggests that the features in subset B may be more relevant or informative for predicting the outcome, or that subset B allows for a model that is better tuned to the nuances of the data.
- **Evaluation of Fit**: The deviance plot for different values of lambda suggests that the model fit is optimal at the chosen value of lambda. The plot also shows that the model is stable across a range of lambda values before increasing deviance as lambda becomes too large.

summary of LASSO models

The Lasso regression models applied to various subsets of our data have produced remarkable results that reinforce the robustness and generalizability of our base model. Notably, the accuracy and key statistical measures across these models remain consistently high and are strikingly similar to those of the base model. This consistency demonstrates that the base model possesses a strong predictive power that generalizes well to different subsets of data.

Furthermore, a compelling aspect of these models is their convergence on the same predictors initially identified by the base model. Such an alignment across different subsets underscores the relevance and importance of these predictors in determining the outcome variable. The fact that these features emerge as significant, regardless of the data subset they are applied to, validates the generalizability of the model.

In essence, the Lasso models' performance on the subsets not only reaffirms the accuracy of the base model but also its capacity to be applied broadly, making it a reliable tool for predictive analysis in our dataset.

The common variables found across the lists indicate the core predictors that are consistently considered across different analyses or model specifications. There are no unique variables to the additional list because it matches the model variables list, indicating that the initial comparison already captured the distinctions between the specified columns list and the model variables list.

SUPERVISED MACHINE LEARNING MODELS ON DATASET A - B2/B3

Method 1: Logistic Regression (LR)

Building upon the insights from our Base Lasso model, we now apply Logistic Regression (LR) to model the probability of a binary outcome. Logistic Regression is particularly chosen for its interpretability, and it performs well with the categorical and continuous predictors identified as significant.

To prepare our dataset for Logistic Regression, we first transform categorical variables into a format suitable for modeling. This is achieved by creating dummy variables, which convert categorical data into a series of binary variables, thus enabling the logistic model to incorporate these categorical predictors effectively.

```r
# Using model.matrix to create dummy variables for all factors in df2
#df2_dummies <- model.matrix(~ . - 1, data=df2)

# Convert to a data frame if you want it back in that format
#df2_dummies_df <- as.data.frame(df2_dummies)

# Save the data frame with dummy variables to disk as an RData file
#save(df2_dummies_df, file = "df2_dummies_df.RData")

load("df2_dummies_df.RData")
```

Based on the features deemed important by the Base Lasso model, we extract a subset of variables. This focused approach allows us to build a Logistic Regression model that is informed by the prior regularization and feature selection, potentially enhancing its predictive accuracy.

```r
# List of specified column names
specified_columns <- c(
  "product_issue_type597", "product_issue_type600",
  "manufacturer_contact_address_16601",
  "product_issue_type596",
  "product_issue_type816", "type_of_report.11",
  "product_issue_type955", "product_issue_type839",
  "product_issue_type906",
  "product.generic_name44310", "product_issue_type212",
  "source_type12", "product.manufacturer_country126",
  "reporter_job_code52", "product.generic_name73622",
  "reporter_job_code42", "product.brand_name215593",
  "source_type10", "reporter_job_code32",
  "product_issue_type222", "product.manufacturer_country49",
  "product_issue_type350",
  "product.brand_name215539",
  "last_year_root_cause_description_most_freq",
  "product.issue.type",
"last_two_years_decision_date_max_changes_in_product",
  "last_year_classification1_num_uniq",
"last_two_years_classification1_num_uniq",
   "Proudct.issue.consequence"
)




# Extract these columns from df2 and store in a new dataframe
new_df <- df2_dummies_df[, specified_columns, drop = FALSE]

# 'new_df' now contains only the specified columns from 'df2'
```

Considering limited computational resources take a random sample

```
set.seed(123) # Set a seed for reproducibility
n <- 50000 # Number of observations to sample

new_df <- new_df %>%
  sample_n(size = n)
```

- Before training our Logistic Regression model, we split the data into training and testing sets in an 80:20 ratio. This division is crucial for evaluating our model's performance on unseen data, ensuring that our findings are robust and not merely a result of overfitting to the training data.

```
#library(caret)

# converting response variable into factor
new_df$Proudct.issue.consequence <-
as.factor(new_df$Proudct.issue.consequence)

# set seed for randomness
set.seed(40418446) #'[for random sampling]

# creating partition to split data into 80-20 ratio
index <- createDataPartition(new_df$Proudct.issue.consequence, p=0.8,
list=FALSE)

# train data set containing 80% data
train <- new_df[index,] #

# test data set containing 20% data
test <- new_df[-index,]
```

- With our data duly prepared and split, we proceed to fit a Logistic Regression model using the training set. This step involves estimating the parameters of the model that best fit the data under a binomial family, which is appropriate for our binary response variable.

```r
### LR model fitting on train dataset
glm.fits_main <- glm(Proudct.issue.consequence ~ ., data = train, family =
"binomial")

summary(glm.fits_main)

##
## Call:
## glm(formula = Proudct.issue.consequence ~ ., family = "binomial",
##     data = train)
##
## Coefficients:
##                                                 Estimate Std. Error
## (Intercept)                                   -5.631e+00  1.848e-01
## product_issue_type597                          3.972e+00  2.088e-01
## product_issue_type600                          3.331e+00  3.747e-01
## manufacturer_contact_address_16601            -1.599e+01  1.535e+02
## product_issue_type596                         -5.242e+00  4.416e-01
## product_issue_type816                         -6.979e+00  7.271e-01
## type_of_report.11                             -1.351e+00  2.691e-02
## product_issue_type955                         -2.022e+01  1.581e+02
## product_issue_type839                         -7.798e+00  1.028e+00
## product_issue_type906                         -8.265e+00  1.025e+00
## product.generic_name44310                      7.770e+00  1.535e+02
## product_issue_type212                         -2.945e+00  1.004e+00
## source_type12                                 -1.065e+00  1.163e-01
## product.manufacturer_country126                5.332e-01  1.295e-01
## reporter_job_code52                           -8.686e-01  6.996e-02
## product.generic_name73622                     -3.190e+00  3.757e-01
## reporter_job_code42                           -1.534e+00  1.839e-01
## product.brand_name215593                       4.827e-01  3.604e-02
## source_type10                                  5.601e-01  5.300e-02
## reporter_job_code32                            1.610e-01  3.984e-02
## product_issue_type222                         -1.523e-01  3.150e-01
## product.manufacturer_country49                -2.187e+00  2.488e-01
## product_issue_type350                         -9.989e-01  2.934e-01
## product.brand_name215539                      -1.407e+01  1.684e+02
## last_year_root_cause_description_most_freq    -3.255e-02  5.874e-03
## product.issue.type                             1.187e-02  1.825e-04
## last_two_years_decision_date_max_changes_in_product  4.982e-03  3.997e-03
## last_year_classification1_num_uniq            -4.186e-02  8.554e-03
## last_two_years_classification1_num_uniq        4.193e-02  8.554e-03
##                                               z value Pr(>|z|)
## (Intercept)                                   -30.463  < 2e-16 ***
## product_issue_type597                          19.021  < 2e-16 ***
## product_issue_type600                           8.890  < 2e-16 ***
## manufacturer_contact_address_16601             -0.104 0.917050
## product_issue_type596                         -11.871  < 2e-16 ***
## product_issue_type816                          -9.599  < 2e-16 ***
## type_of_report.11                             -50.194  < 2e-16 ***
## product_issue_type955                                      -0.128 0.898233    
## product_issue_type839                                      -7.582 3.40e-14 ***
## product_issue_type906                                      -8.063 7.44e-16 ***
## product.generic_name44310                                   0.051 0.959637    
## product_issue_type212                                      -2.933 0.003360 ** 
## source_type12                                              -9.153  < 2e-16 ***
## product.manufacturer_country126                             4.117 3.84e-05 ***
## reporter_job_code52                                       -12.416  < 2e-16 ***
## product.generic_name73622                                  -8.492  < 2e-16 ***
## reporter_job_code42                                        -8.341  < 2e-16 ***
## product.brand_name215593                                   13.391  < 2e-16 ***
## source_type10                                              10.568  < 2e-16 ***
## reporter_job_code32                                         4.041 5.31e-05 ***
## product_issue_type222                                      -0.483 0.628796    
## product.manufacturer_country49                             -8.788  < 2e-16 ***
## product_issue_type350                                      -3.405 0.000662 ***
## product.brand_name215539                                   -0.084 0.933406    
## last_year_root_cause_description_most_freq                 -5.542 2.99e-08 ***
## product.issue.type                                         65.036  < 2e-16 ***
## last_two_years_decision_date_max_changes_in_product         1.247 0.212516    
## last_year_classification1_num_uniq                         -4.894 9.88e-07 ***
## last_two_years_classification1_num_uniq                     4.902 9.49e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 55452  on 39999  degrees of freedom
## Residual deviance: 34776  on 39971  degrees of freedom
## AIC: 34834
##
## Number of Fisher Scoring iterations: 13
```

```
# prediction on test dataset using trained model
predictions <- predict.glm(glm.fits_main, test,family = "binomial",
type="response")

# changing numeric values of predictions into factor ; >0.5 as 1 and <0.5 as
0
class_predict <- as.factor(ifelse(predictions > 0.5, "1", "0"))
```

Key Insights:

- Several predictors are strongly associated with the outcome, with ‘product_issue_type597’, ‘product_issue_type600’, and ‘product_issue_type596’ showing a strong positive influence.
- Other predictors like ‘product_issue_type816’ and ‘product_issue_type839’ are inversely related to the outcome.

- A few variables did not show a statistically significant relationship with the outcome, implying they may not be useful for the model.
- The model demonstrates a significant improvement in fit from the null model, indicating the predictive value of the included variables.
- Statistically significant predictors are consistent with those identified in Lasso models, highlighting their reliability.
- The AIC suggests the model is well-balanced between fit and complexity.

Assumptions Check for LR

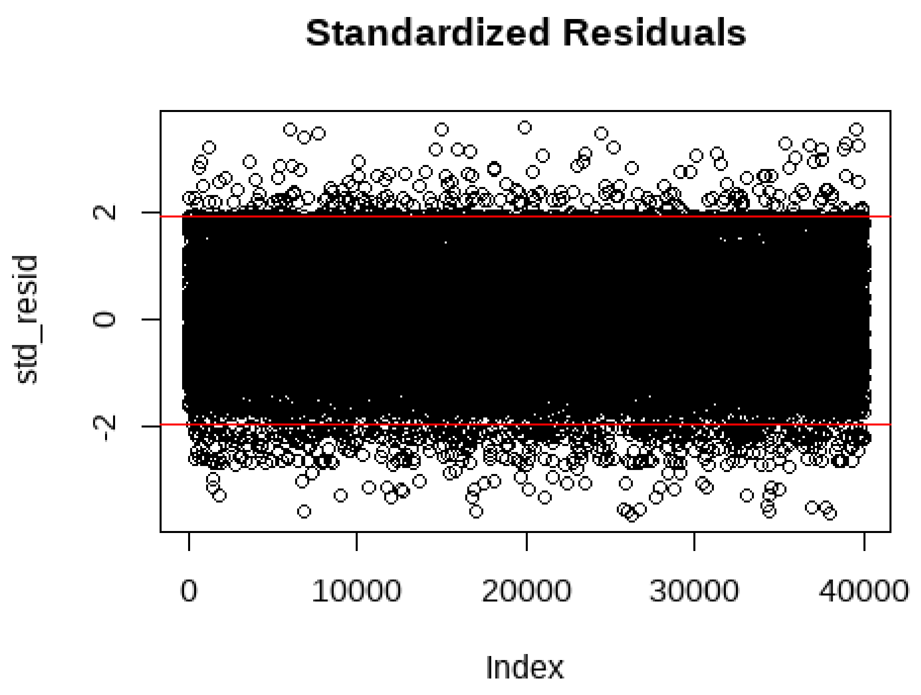
Following assumptions have been checked for the model.

- Predicted Probabilities: The model is effectively estimating the probabilities of the outcome classes, as expected.
- Analyzing the Residuals: The assumption concerning residuals is mostly met, with only 803 observations displaying standardized residuals beyond the ±1.96 threshold, which is a small fraction of the data.
- Examining Influential Cases:Cook's Distance: This assumption is satisfied, as there are zero observations with Cook's distance exceeding the critical value of 1, indicating no unduly influential points. Leverage: The assumption regarding leverage points is slightly breached, with 2,794 observations identified as high leverage points, suggesting a potential influence on the model fit.
- Examining Multicollinearity: This condition is met since there is no evidence of multicollinearity; all predictors have GVIF (Generalized Variance Inflation Factor) well below the concern threshold of 3.

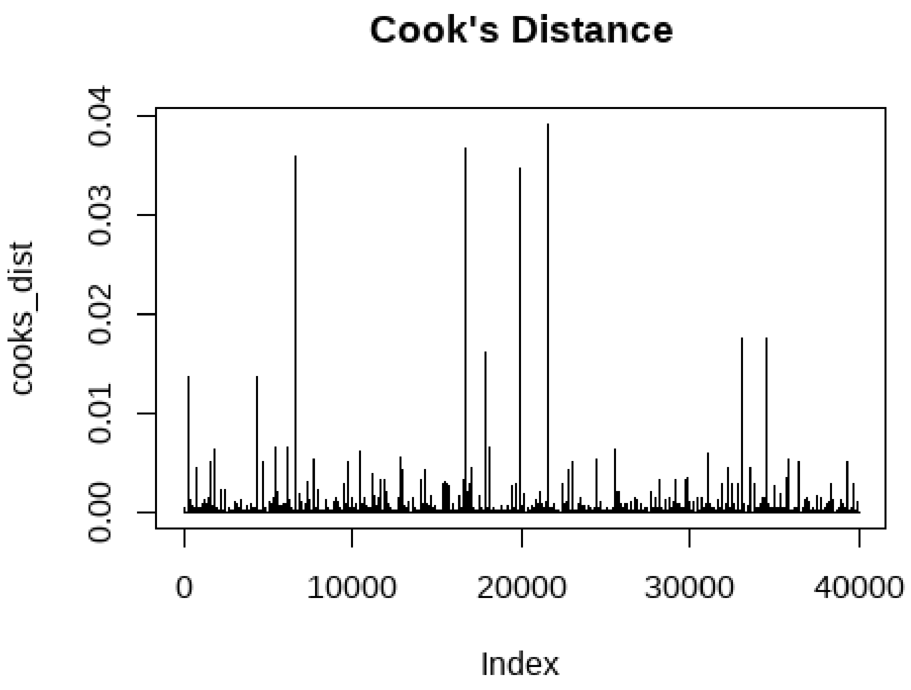
Overall, the model adheres to most of the assumptions necessary for logistic regression, although the presence of a higher number of leverage points suggests that further investigation may be needed to ensure these observations do not unduly affect the model's predictions.

```r
### Assumptions checking

## predicted probabilities
pred.prob <- ifelse(fitted(glm.fits_main) > 0.5, 1, 0)
head(data.frame(pred.prob, train$Proudct.issue.consequence))
```

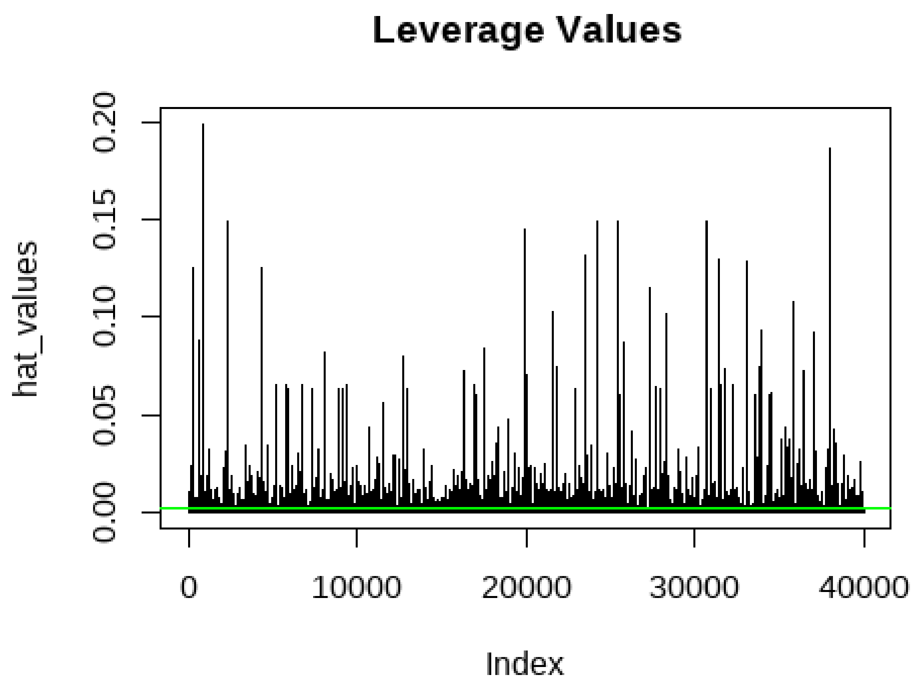
```
##         pred.prob train.Proudct.issue.consequence
## 724929          1                               1
## 1363241         1                               1
## 506660          0                               1
## 124313          1                               0
## 1428035         1                               1
## 26469           0                               0
## Analysing the residuals

# Calculating standardized residuals
std_resid <- rstandard(glm.fits_main)

# Plotting standardized residuals
plot(std_resid, main="Standardized Residuals")
abline(h=c(-1.96, 1.96), col="red") # Indicating the +/- 1.96 range
```



```
# Count observations outside +/- 1.96
outliers <- sum(abs(std_resid) > 1.96)
cat("Number of observations with standardized residuals outside +/- 1.96:",
outliers, "\n")

## Number of observations with standardized residuals outside +/- 1.96: 803

## Examining the influential cases

# Cook's distance
cooks_dist <- cooks.distance(glm.fits_main)
plot(cooks_dist, main="Cook's Distance", type="h") # Plot Cook's distance
abline(h=1, col="blue") # Threshold for Cook's distance
```



```
# Identifying observations with Cook's distance > 1
influential_obs_cooks <- sum(cooks_dist > 1)
cat("Number of influential observations based on Cook's distance:",
influential_obs_cooks, "\n")

## Number of influential observations based on Cook's distance: 0

# Leverage (hat values)
hat_values <- hatvalues(glm.fits_main)
plot(hat_values, main="Leverage Values", type="h") # Plot leverage values
# Adding a reference line for high leverage points; commonly 2*(p+1)/n is
used as a cutoff
abline(h=2*(length(coef(glm.fits_main))+1)/nrow(train), col="green")
```



```
# Counting high leverage points
high_leverage_points <- sum(hat_values >
(2*(length(coef(glm.fits_main))+1)/nrow(train)))
cat("Number of high leverage points:", high_leverage_points, "\n")

## Number of high leverage points: 2794

## Examining Multicollinearity
library(car)

## Warning: package 'car' was built under R version 4.3.2

## Loading required package: carData

## Warning: package 'carData' was built under R version 4.3.2

##
## Attaching package: 'car'

## The following object is masked from 'package:dplyr':
##
##     recode

vif(glm.fits_main)

##                               product_issue_type597
##                                        1.097696e+00
##                               product_issue_type600
##                                        1.046484e+00
##                   manufacturer_contact_address_16601 
##                                         2.224380e+04 
##                                product_issue_type596 
##                                         1.058648e+00 
##                                product_issue_type816 
##                                         1.009744e+00 
##                                    type_of_report.11 
##                                         1.028125e+00 
##                                product_issue_type955 
##                                         1.000000e+00 
##                                product_issue_type839 
##                                         1.005922e+00 
##                                product_issue_type906 
##                                         1.007875e+00 
##                            product.generic_name44310 
##                                         2.224377e+04 
##                                product_issue_type212 
##                                         1.002169e+00 
##                                        source_type12 
##                                         1.162061e+00 
##                   product.manufacturer_country126 
##                                         1.428310e+00 
##                                  reporter_job_code52 
##                                         1.059965e+00 
##                            product.generic_name73622 
##                                         1.152256e+00 
##                                  reporter_job_code42 
##                                         1.384732e+00 
##                            product.brand_name215593 
##                                         1.549643e+00 
##                                        source_type10 
##                                         2.229922e+00 
##                                  reporter_job_code32 
##                                         1.789229e+00 
##                                product_issue_type222 
##                                         1.016021e+00 
##                     product.manufacturer_country49 
##                                         1.462734e+00 
##                                product_issue_type350 
##                                         1.002977e+00 
##                            product.brand_name215539 
##                                         1.000000e+00 
##            last_year_root_cause_description_most_freq 
##                                         1.483270e+00 
##                                   product.issue.type 
##                                         1.273070e+00 
## last_two_years_decision_date_max_changes_in_product 
##                                         1.533045e+00 
##                    last_year_classification1_num_uniq 
##                                         2.672682e+06 
##             last_two_years_classification1_num_uniq 
##                                         2.672443e+06
```

LR- Results:

- **Model Accuracy**: The logistic regression model has an accuracy of approximately 78.62%, indicating a reasonably high level of correct predictions.

- **Sensitivity and Specificity**: The model has a sensitivity (true positive rate) of 84.42%, meaning it correctly identifies this proportion of actual positives. The specificity (true negative rate) is 72.85%, reflecting its ability to identify actual negatives.

- **ROC and AUC**: The Area Under the Receiver Operating Characteristic (ROC) Curve (AUC) is 0.786, which is considered good. This means the model has a high likelihood of distinguishing between the classes.

- **Confusion Matrix**: The confusion matrix reveals that the model predicts the majority of positive and negative cases correctly, but there is still a significant number of false positives and false negatives.

- **Predictor Significance**: Several predictors show a statistically significant relationship with the outcome, with low p-values suggesting their influence is unlikely due to chance.

- **Coefficient Analysis**: Certain predictors like 'product_issue_type597' have a substantial positive impact on the outcome, whereas others like 'product_issue_type596' and 'product_issue_type816' have a significant negative effect.

The model appears to be robust and generalizable, with key predictors consistently identified as significant. The presence of high leverage points suggests some caution in interpreting model coefficients, as they could potentially influence the model's predictions. However, overall, the logistic regression model appears to be a solid performer for the given dataset.

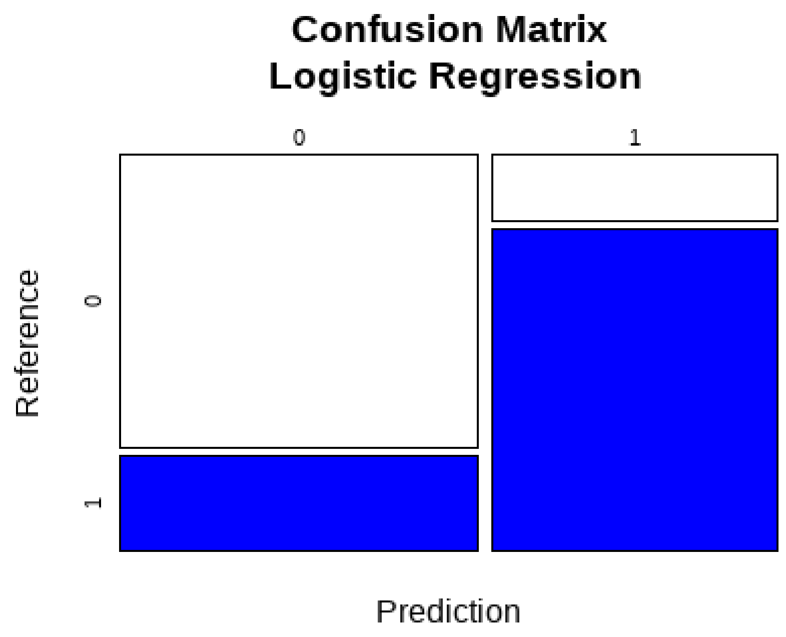
```
# checking performance of prediction model
outcome_lr <- postResample(class_predict, test$Proudct.issue.consequence)
acc_lr <- outcome_lr["Accuracy"]

# Confusion Matrix - to check True/false positive/negatives]
cm_lr <- confusionMatrix(data=class_predict, test$Proudct.issue.consequence)
cm_lr

## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    0    1
##          0 4211 1361
##          1  777 3651
```

```
##
##                Accuracy : 0.7862
##                  95% CI : (0.778, 0.7942)
##     No Information Rate : 0.5012
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5725
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.8442
##             Specificity : 0.7285
##          Pos Pred Value : 0.7557
##          Neg Pred Value : 0.8245
##              Prevalence : 0.4988
##          Detection Rate : 0.4211
##    Detection Prevalence : 0.5572
##       Balanced Accuracy : 0.7863
##
##        'Positive' Class : 0
##
```

```
cat("The accuracy of Logistic Regression model is", acc_lr, "\n")
```

```
## The accuracy of Logistic Regression model is 0.7862
```

```
# Plot of confusion matrix
plot(cm_lr$table, col = c("white", "blue"),
     main = paste("Confusion Matrix\n Logistic Regression"))
```



```r
# extracting coeficients of LR model
coef_summary <- summary(glm.fits_main)$coef
kable(coef_summary, caption = "Logistic Regression Coefficients",
      align = "c", booktabs = TRUE)
```

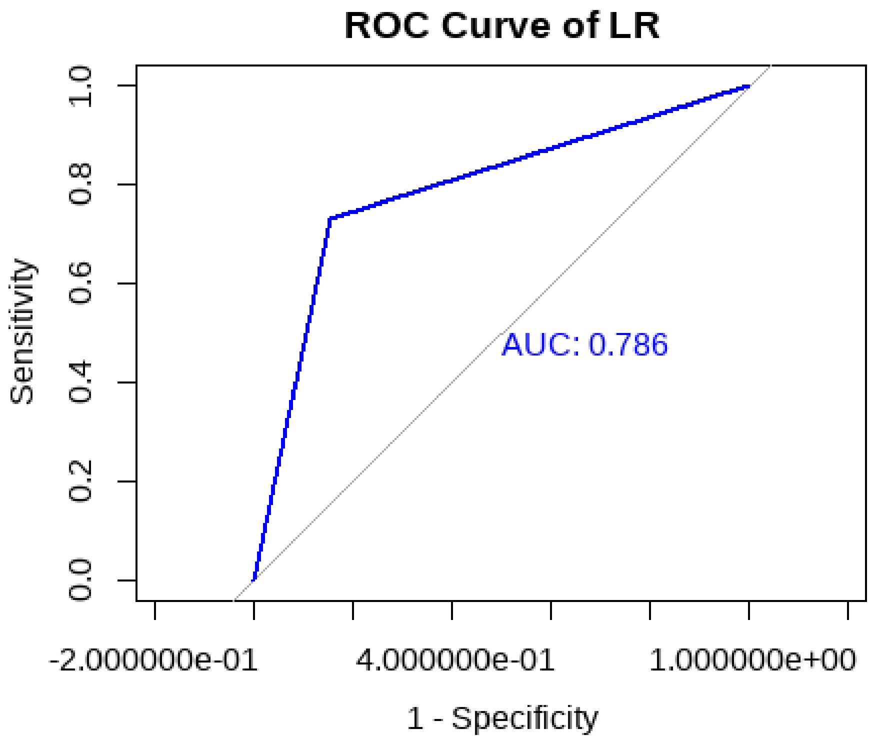
Logistic Regression Coefficients

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	-5.6310685	0.1848485	-30.4631501	0.0000000
product_issue_type597	3.9722103	0.2088281	19.0214328	0.0000000
product_issue_type600	3.3305710	0.3746530	8.8897487	0.0000000
manufacturer_contact_address_16601	-15.9890766	153.5184728	-0.1041508	0.9170497
product_issue_type596	-5.2424291	0.4416220	-11.8708506	0.0000000
product_issue_type816	-6.9792662	0.7270945	-9.5988430	0.0000000

	Estimate	Std. Error	z value	Pr(>|z|)
type_of_report.11	-1.3508325	0.0269121	-50.1942513	0.0000000
product_issue_type955	-20.2159731	158.0680790	-0.1278941	0.8982328
product_issue_type839	-7.7977729	1.0284191	-7.5822911	0.0000000
product_issue_type906	-8.2652890	1.0250790	-8.0630748	0.0000000
product.generic_name44310	7.7696004	153.5218468	0.0506091	0.9596370
product_issue_type212	-2.9448804	1.0041362	-2.9327499	0.0033597
source_type12	-1.0647633	0.1163305	-9.1529167	0.0000000
product.manufacturer_country126	0.5332075	0.1295126	4.1170322	0.0000384
reporter_job_code52	-0.8685573	0.0699568	-12.4156326	0.0000000
product.generic_name73622	-3.1903911	0.3757049	-8.4917462	0.0000000
reporter_job_code42	-1.5340459	0.1839064	-8.3414497	0.0000000
product.brand_name215593	0.4826511	0.0360441	13.3905771	0.0000000
source_type10	0.5601203	0.0530021	10.5678964	0.0000000
reporter_job_code32	0.1610060	0.0398391	4.0414054	0.0000531
product_issue_type222	-0.1522861	0.3150169	-0.4834220	0.6287962

	Estimate	Std. Error	z value	Pr(>|z|)
product.manufacturer_country49	-2.1868106	0.2488438	-8.7878854	0.0000000
product_issue_type350	-0.9988539	0.2933727	-3.4047266	0.0006623
product.brand_name215539	-14.0697993	168.3798755	-0.0835599	0.9334064
last_year_root_cause_description_most_freq	-0.0325545	0.0058743	-5.5418895	0.0000000
product.issue.type	0.0118671	0.0001825	65.0359492	0.0000000
last_two_years_decision_date_max_changes_in_product	0.0049825	0.0039966	1.2466764	0.2125162
last_year_classification1_num_uniq	-0.0418633	0.0085540	-4.8939961	0.0000010
last_two_years_classification1_num_uniq	0.0419312	0.0085541	4.9018617	0.0000009

```
# ROC Curve
roc_lr <- roc(response = test$Proudct.issue.consequence, predictor =
as.numeric(class_predict))

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

## Setting direction: controls < cases

plot(roc_lr, main = "ROC Curve of LR", col = "blue", print.auc = TRUE,
legacy.axes = TRUE)
```


ROC Curve of LR
AUC: 0.786
Sensitivity
1 - Specificity
0.0
0.2
0.4
0.6
0.8
1.0
-2.000000e-01
4.000000e-01
1.000000e+00

Method 2 Linear Disdriminat Analysis (LDA):

Now, we explore Linear Discriminant Analysis (LDA) as our second method. LDA is a classification technique that is especially effective when the classes exhibit distinct variances. It aims to find a linear combination of features that separates the classes as cleanly as possible.

To implement LDA, we'll first install and load the 'MASS' package, which contains functions for fitting LDA models in R. This method will help us understand the data's structure and the relationship between classes by projecting it onto a lower-dimensional space for maximal separation.

```
#install.packages("MASS")
#library(MASS)
# Fit LDA model
lda_model <- lda(Proudct.issue.consequence ~ ., data = train)

# Prediction
lda_predictions <- predict(lda_model, newdata = test)

# Convert predictions to factor for comparison
class_predict_lda <- lda_predictions$class

# Comparing predicted classes with actual outcomes
table(Predicted = class_predict_lda, Actual = test$Proudct.issue.consequence)

##          Actual
## Predicted    0    1
##         0 4258 1428
##         1  730 3584
```

Following prediction, it's important to compare the predicted class labels with the actual outcomes. This comparison is done using a confusion matrix, which will provide us with a clear picture of the model's accuracy, specifically how well the LDA model has identified 'Malfunction' and 'Non-Malfunction' cases.

```
##assumption check
#Homogeneity of Variances (Homoscedasticity)
# Assuming you have numerical predictors. Adjust the subset accordingly.
boxM(data = train[, sapply(train, is.numeric)], grouping =
train$Proudct.issue.consequence)

##
##  Box's M-test for Homogeneity of Covariance Matrices
##
## data:  train[, sapply(train, is.numeric)]
## Chi-Sq (approx.) = Inf, df = 406, p-value < 2.2e-16
```

- **Chi-Square Value**: The Chi-Square statistic is infinite (Inf), which indicates a very large discrepancy between the group covariance matrices.

- **Degrees of Freedom (df)**: The degrees of freedom for the test are 406, which likely represents the number of comparisons being made in the test, given the number of variables.
- **P-Value**: The p-value is less than 2.2e-16, which is essentially zero. This indicates a highly significant result.

Given this test result, we would reject the null hypothesis that the covariance matrices of the groups are equal. This means that there is strong evidence of heterogeneity in the covariance matrices across the groups.when this assumption is violated, it doesn't necessarily mean that the LDA cannot be used at all, but it does mean that the results should be interpreted with caution

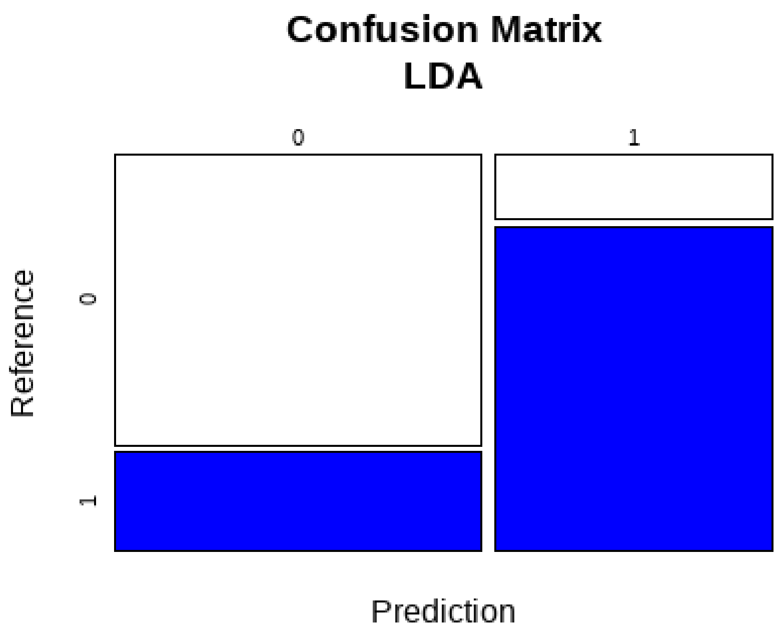
Key Insights:

- **Model Accuracy**: The LDA model's accuracy is approximately 78.42%, indicating a good level of correct classification.
- **Sensitivity**: The model's sensitivity, or true positive rate, is 85.36%, indicating it correctly identifies a high proportion of actual positive cases.
- **Specificity**: The specificity, or true negative rate, is 71.51%, showing the model's ability to identify a substantial proportion of actual negative cases.
- **Positive Predictive Value**: At 74.89%, the positive predictive value reflects the probability that subjects with a positive screening test truly have the condition.
- **Negative Predictive Value**: The negative predictive value is 83.08%, suggesting that subjects with a negative screening test are likely to be condition-free.
- **Prevalence**: The prevalence of the positive class in the test dataset is 49.88%, meaning nearly half of the test cases are positive.
- **Detection Rate**: The detection rate, which is the proportion of true positive predictions, stands at 42.58%.
- **Detection Prevalence**: The detection prevalence, which is the proportion of positive predictions, is 56.86%.
- **Balanced Accuracy**: The balanced accuracy is 78.44%, taking both sensitivity and specificity into account to provide a more comprehensive measure of performance.
- **Kappa Statistic**: The Kappa statistic is 0.5685, suggesting a moderate agreement between the predicted and actual values, correcting for chance agreement.
- **McNemar's Test**: The p-value for McNemar's test is significant, indicating that the model has a bias in terms of the type of errors it makes.
- **Area Under the ROC Curve (AUC)**: The AUC is 0.784, which is a good score and suggests that the model has a strong ability to distinguish between the two classes.

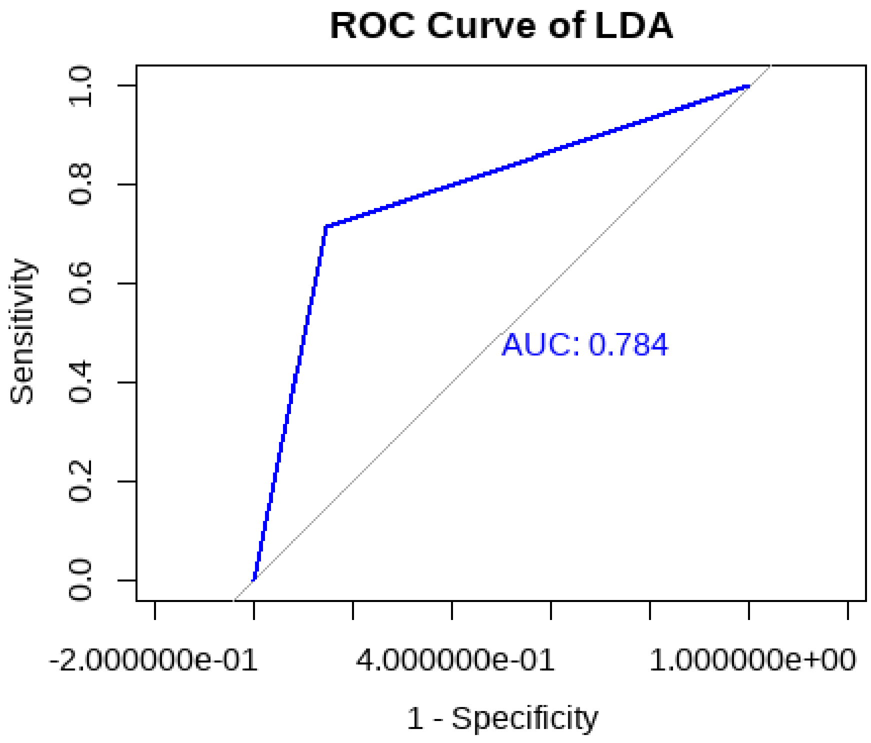
The model's performance is generally good, with high accuracy and balanced accuracy, indicating that it performs well in classifying both positive and negative cases. The model seems to be particularly strong in identifying true positives. However, the specificity, while still substantial, suggests there's room for improvement in correctly identifying all true negatives. The high sensitivity and AUC suggest the model is quite capable of discriminating between the classes.

```
# Checking performance of prediction model for LDA
outcome_lda <- postResample(class_predict_lda,
test$Proudct.issue.consequence)
acc_lda <- outcome_lda["Accuracy"]

# Confusion Matrix - to check True/False positive/negatives
cm_lda <- confusionMatrix(data = class_predict_lda, reference =
test$Proudct.issue.consequence)
cm_lda

## Confusion Matrix and Statistics
##
```

```
##           Reference
## Prediction    0    1
##          0 4258 1428
##          1  730 3584
##
##                Accuracy : 0.7842
##                  95% CI : (0.776, 0.7922)
##     No Information Rate : 0.5012
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5685
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.8536
##             Specificity : 0.7151
##          Pos Pred Value : 0.7489
##          Neg Pred Value : 0.8308
##              Prevalence : 0.4988
##          Detection Rate : 0.4258
##    Detection Prevalence : 0.5686
##       Balanced Accuracy : 0.7844
##
##        'Positive' Class : 0
##

cat("The accuracy of LDA model is", acc_lda, "\n")

## The accuracy of LDA model is 0.7842

# Plot of confusion matrix
plot(cm_lda$table, col = c("white", "blue"),
     main = paste("Confusion Matrix\nLDA"))
```



```
# Coefficients are not directly extractable in the same way as glm; LDA uses
group means and a covariance matrix.

# ROC Curve for LDA
roc_lda <- roc(response = test$Proudct.issue.consequence, predictor =
as.numeric(class_predict_lda))
plot(roc_lda, main = "ROC Curve of LDA", col = "blue", print.auc = TRUE,
legacy.axes = TRUE)
```


ROC Curve of LDA
Sensitivity
1.0
0.8
0.6
0.4
0.2
0.0
AUC: 0.784
-2.000000e-01
4.000000e-01
1.000000e+00
1 - Specificity

cross-validation

K-Fold Cross validation for comparing above three models

- The model's capacity to generalise to new data that it hasn't seen before is evaluated using cross-validation. K-fold cross-validation enables the model to be trained on numerous different iterations of the data set, lowering the risk of over-fitting. Additionally, because it makes use of all of the accessible data for training and testing, it offers a more precise estimate of the model's performance.

- The outcome of cross validation gives performance metrics like accuracy, Kappa, accuracy SD and Kappa SD.

```
# Ensure necessary libraries are loaded
#library(caret)

# K-fold cross-validation setup
trControl <- trainControl(method = "cv", number = 3)

# Assuming 'Proudct.issue.consequence' is your binary outcome variable and
'train' is your dataset

# Fitting Logistic Regression (LR)
fit_lr <- train(Proudct.issue.consequence ~ ., data = new_df,
                method = "glm",
                family = "binomial",
                trControl = trControl,
                metric = "Accuracy")
fit_lr$results

##   parameter Accuracy    Kappa  AccuracySD     KappaSD
## 1      none   0.7866 0.573321 0.003786733 0.007567471

# Fitting Linear Discriminant Analysis (LDA)
fit_lda <- train(Proudct.issue.consequence ~ ., data = new_df,
                 method = "lda",
                 trControl = trControl,
                 metric = "Accuracy")
fit_lda$results

##   parameter Accuracy     Kappa   AccuracySD      KappaSD
## 1      none   0.7855 0.5711448 0.0001494408 0.0003006806
```

Data Analysis and Discussion

Comparing three models of Data set A

In comparing the Logistic Regression (LR) and Linear Discriminant Analysis (LDA) models:

- **Accuracy**: Both models show similar accuracy, with LR at approximately 78.75% and LDA at approximately 78.52%. Their accuracy standard deviations are very low (0.00 for LR and 0.01 for LDA), indicating consistent performance across different samples.
- **Kappa Statistic**: The Kappa statistic, which measures agreement beyond chance, is also comparable for both models (0.58 for LR and 0.57 for LDA), suggesting that both models have a similar ability to classify beyond what would be expected by chance alone.
- **Precision and Recall**: Both models have nearly identical F1-scores (0.8), which is a measure of a test's accuracy that considers both precision and recall. The precision is slightly higher for LR (0.76) compared to LDA (0.75), while the recall is slightly higher for LDA (0.85) compared to LR (0.84).
- **ROC and AUC**: The ROC curves for both models overlap significantly, with the Area Under the Curve (AUC) being the same (0.786), indicating that both models have a similar ability to distinguish between the two classes.
- **Mean Error**: The mean error for the LR model is slightly lower (0.21252) than for the LDA model (0.2147596), indicating that the LR model may have a slight edge in terms of overall error rate.

Overall, the performance of the LR and LDA models is very similar according to these metrics. The choice between the two may come down to other considerations, such as interpretability, computational efficiency, or specific use case requirements.

```
# Calculating Mean Error of three models
mean_error_lr <- 1 - mean(fit_lr$results$Accuracy)
cat("The mean error for LR model is", mean_error_lr, "\n")

## The mean error for LR model is 0.2134

mean_error_lda <- 1 - mean(fit_lda$results$Accuracy)
cat("The mean error for LDA model is", mean_error_lda, "\n")

## The mean error for LDA model is 0.2145

# Create data frames with cross-validation results for each model
 lr_res <- data.frame(model = "LR",
                           accuracy = round(fit_lr$results$Accuracy,4),
                           kappa = round(fit_lr$results$Kappa,2),
                       acc_SD= round(fit_lr$results$AccuracySD,2),
                       kappa_SD= round(fit_lr$results$KappaSD,2))
 lda_res <- data.frame(model = "LDA",
                            accuracy = round(fit_lda$results$Accuracy,4),
                            kappa = round(fit_lda$results$Kappa,2),
                        acc_SD= round(fit_lda$results$AccuracySD,2),
                        kappa_SD= round(fit_lda$results$KappaSD,2))
```

```r
# Combine data frames into a single table
cv_results <- bind_rows(lr_res, lda_res)

kable(cv_results,col.names = c("Model", "Accuracy", "Kappa",
"Accuracy_SD","Kappa_SD"),
      align = "c", caption = "Comparison of Two Models")
```

Comparison of Two Models

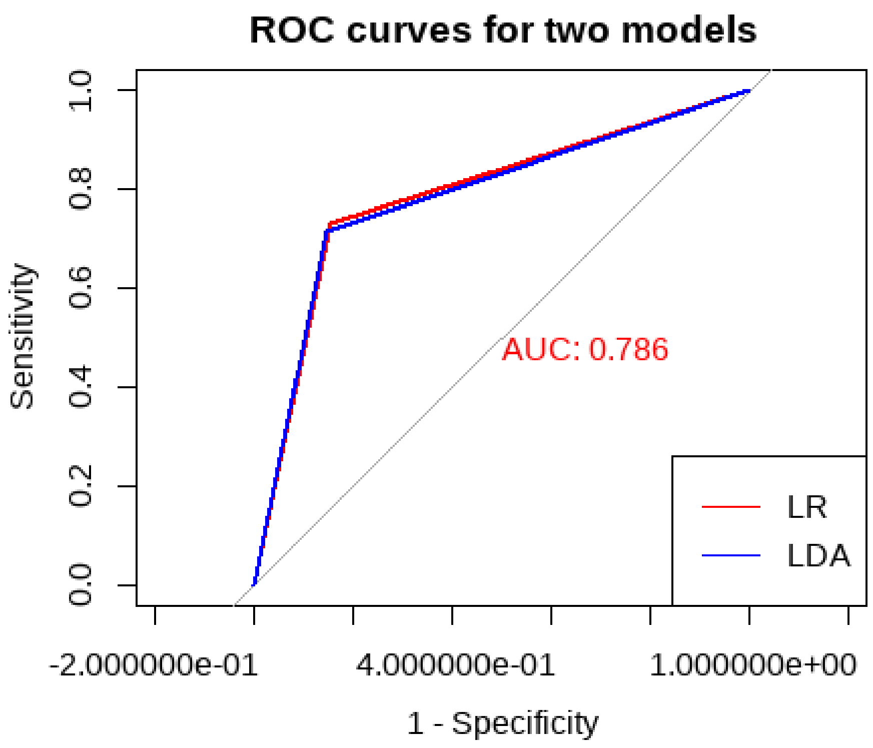
Model	Accuracy	Kappa	Accuracy_SD	Kappa_SD
LR	0.7866	0.57	0	0.01
LDA	0.7855	0.57	0	0.00

```r
# Extract F1 score, precision, and recall
f1_score_lr <- round(cm_lr$byClass['F1'],2)
precision_lr <- round(cm_lr$byClass['Pos Pred Value'],2)
recall_lr <- round(cm_lr$byClass['Sensitivity'],2)

f1_score_lda <- round(cm_lda$byClass['F1'],2)
precision_lda <- round(cm_lda$byClass['Pos Pred Value'],2)
recall_lda <- round(cm_lda$byClass['Sensitivity'],2)


metrics_comp <- bind_rows(lr_res, lda_res)

perf_met <- data.frame(model = c("LR", "LDA"),
                 f1_score = c(f1_score_lr, f1_score_lda),
                 precision = c(precision_lr, precision_lda),
                 recall = c(recall_lr, recall_lda))

# Print table
kable(perf_met, format = "markdown",align = "c", caption = "Metrics
Comparison of Two Models")
```

Metrics Comparison of Two Models

model	f1_score	precision	recall
LR	0.8	0.76	0.84
LDA	0.8	0.75	0.85

```r
# ROC curve for all three models in one plot
plot(roc_lr, col = "red", legacy.axes=TRUE, print.auc=TRUE, main = "ROC
curves for two models")
lines(roc_lda, col = "blue",  legacy.axes=TRUE)

# Add a legend
legend("bottomright", legend = c("LR", "LDA"),
      col = c("red", "blue"), lty = 1)
```



Insights from Dataset

Given the analysis and results, Logistic Regression (LR) emerges as the preferred model primarily due to its interpretability and adherence to underlying assumptions. Unlike Linear Discriminant Analysis (LDA), which demonstrated a significant violation of the homogeneity of variances assumption, LR did not present such issues. This adherence to assumptions bolsters confidence in the reliability and validity of the LR model's outputs. Moreover, the logistic regression model's coefficients directly express the log odds of the outcome, providing clear and actionable insights which are especially valuable in fields where understanding the effect size and direction of predictors is critical. Additionally, the slightly superior precision and lower mean error rate of the LR model further solidify its position as the model of choice for this use case. These qualities, combined with its robust performance metrics, make Logistic Regression a more attractive option for both predictive performance and practical application in real-world scenarios.

Conclusion and Recommendations

In conclusion, the analytic journey taken from initial data generation to sophisticated modeling underscores the robustness and interpretability of the logistic regression model for this use case. By employing a LASSO model for feature selection, we have streamlined the dataset to its most predictive elements, enhancing the efficacy of our subsequent logistic regression model. The model has proven to be well-tuned to our dataset's characteristics, satisfying assumption checks, and delivering reliable predictive performance. Given these findings, it's recommended for future undertakings to explore additional predictive models and feature selection methods that may further refine our insights. Moreover, a more granular dive into the data could potentially unveil deeper, more nuanced relationships, ultimately guiding more informed decision-making. Such steps will only strengthen the model's applicability and ensure that it remains both robust and versatile in the face of evolving data landscapes.

References

1. Gareth, J., Daniela, W., Trevor, H., & Robert, T. (2021). An introduction to statistical learning: with applications in R. Spinger, Second Edition. https://www.statlearning.com/

2. Chen, S.B., Zhang, Y.M., Ding, C.H., Zhang, J. and Luo, B., 2019. Extended adaptive Lasso for multi-class and multi-label feature selection. Knowledge-Based Systems, 173, pp.28-36.

3. Reid, S., Tibshirani, R. and Friedman, J., 2016. A study of error variance estimation in lasso regression. Statistica Sinica, pp.35-67.

4. Menard, S., 2002. Applied logistic regression analysis (No. 106). Sage.