Healthcare status prediction: A Supervised machine learning approach

Adekunle Joseph Damilare

2024-04-13

```
library(tidyverse)
```

```
## Warning: package 'ggplot2' was built under R version 4.3.3
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
## ✓ dplyr     1.1.3     ✓ readr     2.1.4
## ✓ forcats   1.0.0     ✓ stringr   1.5.0
## ✓ ggplot2   3.5.0     ✓ tibble    3.2.1
## ✓ lubridate 1.9.3     ✓ tidyr     1.3.0
## ✓ purrr     1.0.2
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to becom
e errors
```

Accessing the dataset

```
url1 <- "https://raw.githubusercontent.com/josephdamilare01/Data-Mining-Data-Mining-the-Healthca
re-Dataset/main/disease_train(5).csv"
```

```
Train <- read.csv(url1)
head(Train)
```

	id <chr>	age <int>	gender <chr>	sick <chr>	pregnant <chr>	test_X1 <dbl>	test_X2 <dbl>	test_X3 <dbl>	test_X4 <dbl>
1	PA1001	59	male	no	no	7.8	*NA*	89	0.85
2	PA1002	48	female	no	no	1.5	2.5	101	0.97
3	PA1003	77	male	no	no	7.3	1.2	57	1.28
4	PA1004	42	female	no	no	1.2	2.5	106	0.98
5	PA1005	38	female	no	no	0.6	1.9	95	*NA*
6	PA1006	44	male	no	no	3.0	2.0	115	1.10

6 rows | 1-10 of 25 columns

Data exploration

```
str(Train)
```

```
## 'data.frame':    4250 obs. of  24 variables:
##  $ id             : chr  "PA1001" "PA1002" "PA1003" "PA1004" ...
##  $ age            : int  59 48 77 42 38 44 90 69 33 42 ...
##  $ gender         : chr  "male" "female" "male" "female" ...
##  $ sick           : chr  "no" "no" "no" "no" ...
##  $ pregnant       : chr  "no" "no" "no" "no" ...
##  $ test_X1        : num  7.8 1.5 7.3 1.2 0.6 3 1.5 6.9 0.1 1.9 ...
##  $ test_X2        : num  NA 2.5 1.2 2.5 1.9 2 1.8 NA 1.7 2.2 ...
##  $ test_X3        : num  89 101 57 106 95 115 98 109 104 126 ...
##  $ test_X4        : num  0.85 0.97 1.28 0.98 NA 1.1 0.94 1.03 0.8 0.97 ...
##  $ test_X5        : num  105 104 44 108 NA 104 104 106 130 130 ...
##  $ test_X6        : num  NA NA NA 27 NA NA NA NA NA NA ...
##  $ concern_type1  : chr  "no" "no" "no" "no" ...
##  $ concern_type2  : chr  "yes" "no" "no" "no" ...
##  $ enlargement    : chr  "no" "no" "no" "no" ...
##  $ tumor          : chr  "no" "no" "no" "no" ...
##  $ disorder       : chr  "no" "no" "no" "no" ...
##  $ medication_A   : chr  "no" "yes" "no" "no" ...
##  $ medication_B   : chr  "no" "no" "no" "no" ...
##  $ mental_health  : chr  "no" "no" "no" "no" ...
##  $ mood_stabiliser: chr  "no" "yes" "no" "no" ...
##  $ surgery        : chr  "no" "no" "no" "no" ...
##  $ treatment_type1: chr  "no" "no" "no" "no" ...
##  $ suspect        : chr  "no" "no" "no" "no" ...
##  $ target         : chr  "moderate_risk" "low_risk" "moderate_risk" "low_risk" ...
```

Checking for missing values using dlookr package

```
library(dlookr)
```

```
## Warning: package 'dlookr' was built under R version 4.4.0
```

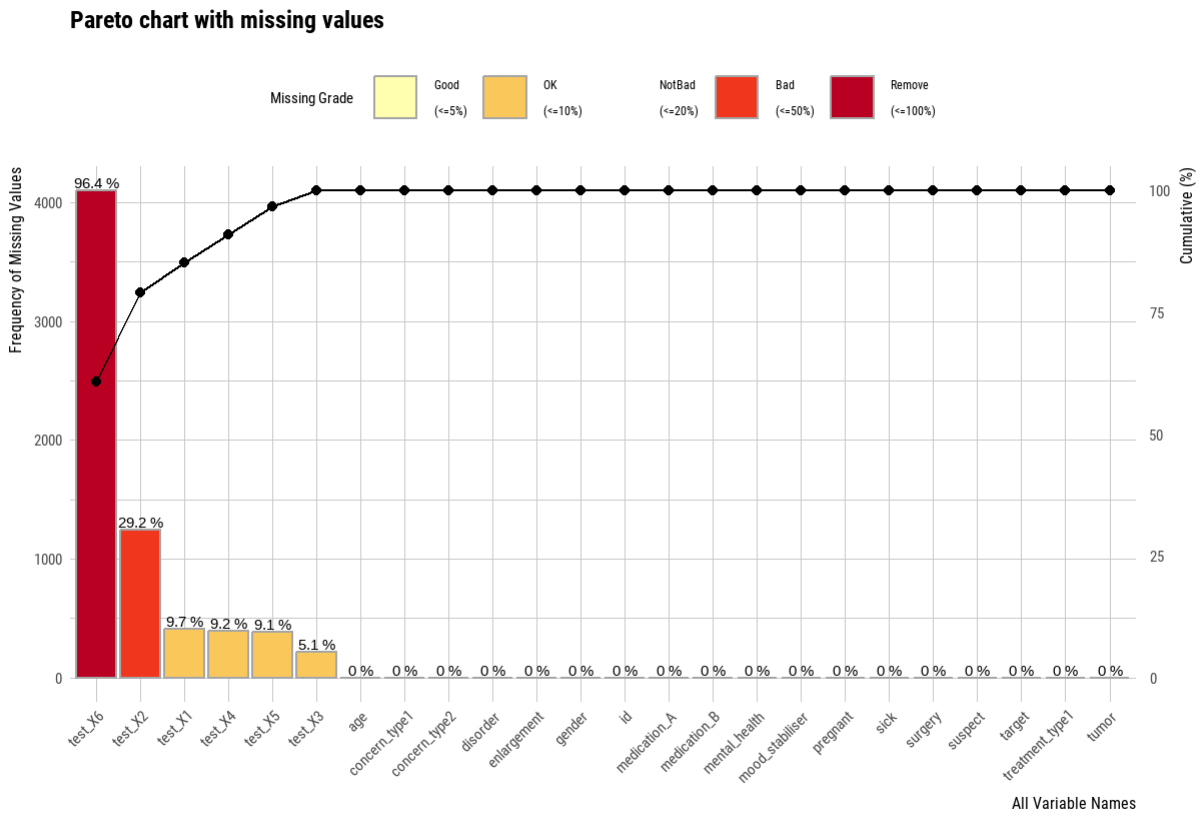
```
## Registered S3 methods overwritten by 'dlookr':
##   method          from
##   plot.transform  scales
##   print.transform scales
```

```
##
## Attaching package: 'dlookr'
```

```
## The following object is masked from 'package:tidyr':
##
##     extract
```

```
## The following object is masked from 'package:base':
##
##     transform
```

```
plot_na_pareto(Train)
```



Checking for outliers

```
diagnose_outlier(Train)
```

variables <chr>	outliers_cnt <int>	outliers_ratio <dbl>	outliers_mean <dbl>	with_mean <dbl>	without_mean <dbl>
age	2	0.04705882	32990.500000	67.374824	51.8742938
test_X1	480	11.29411765	47.657708	7.342463	1.5814275
test_X2	196	4.61176471	3.393878	2.035580	1.9408716

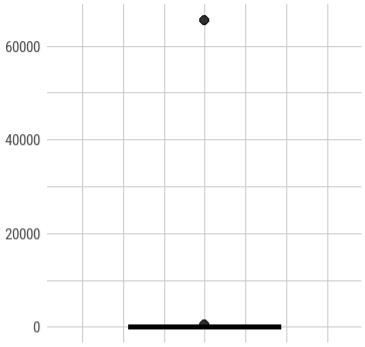
variables <chr>	outliers_cnt <int>	outliers_ratio <dbl>	outliers_mean <dbl>	with_mean <dbl>	without_mean <dbl>
test_X3	224	5.27058824	123.811429	104.919623	103.8089239
test_X4	112	2.63529412	1.244911	0.970846	0.9626519
test_X5	238	5.60000000	136.907101	110.090834	108.3302069
test_X6	4	0.09411765	25.050000	23.325974	23.2800000

7 rows

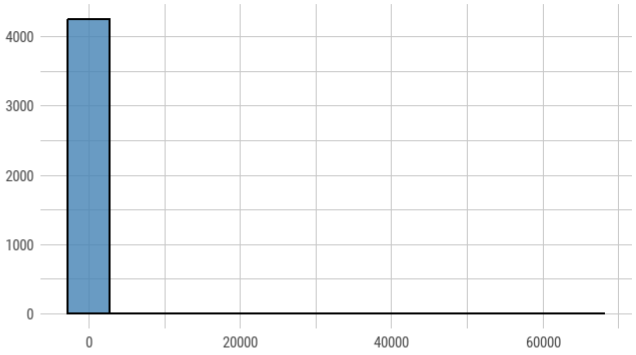
```
plot_outlier(Train)
```

Outlier Diagnosis Plot (age)

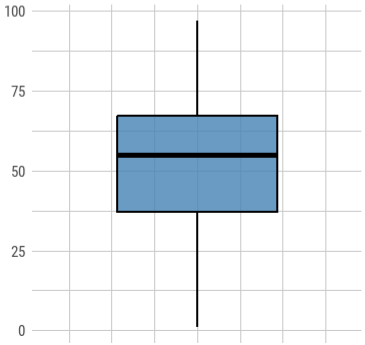
With outliers



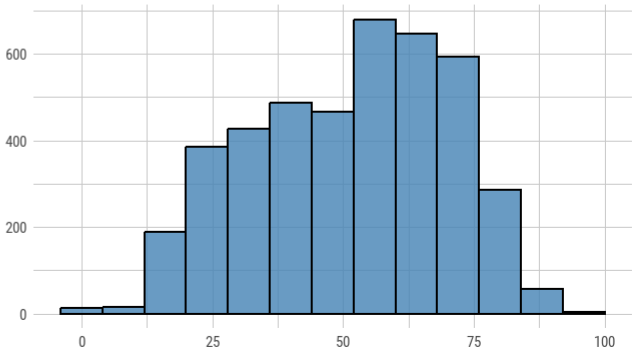
With outliers



Without outliers

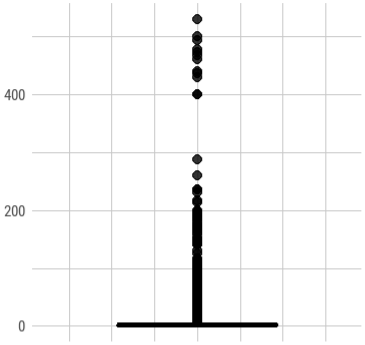


Without outliers

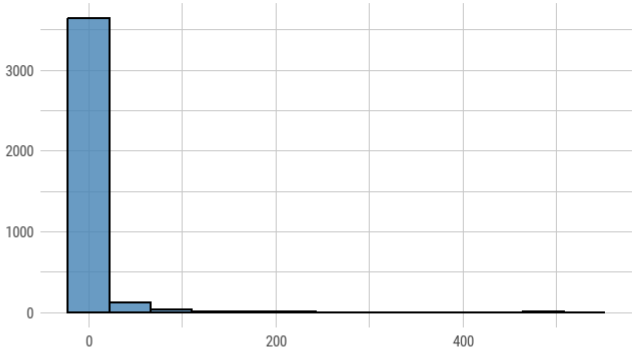


Outlier Diagnosis Plot (test_X1)

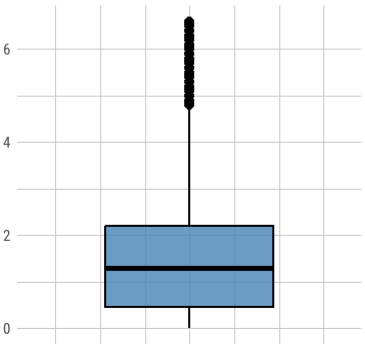
With outliers



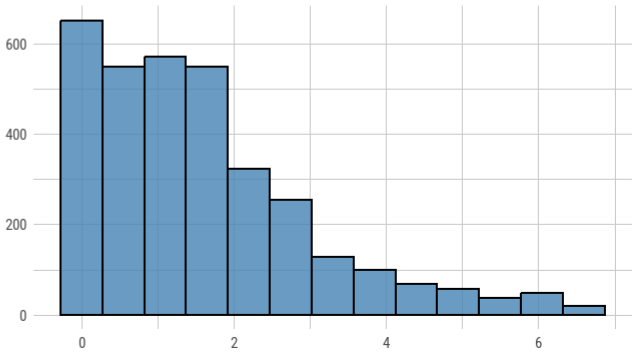
With outliers



Without outliers

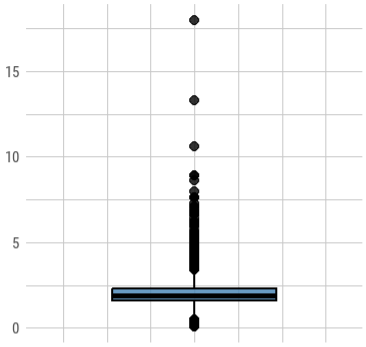


Without outliers

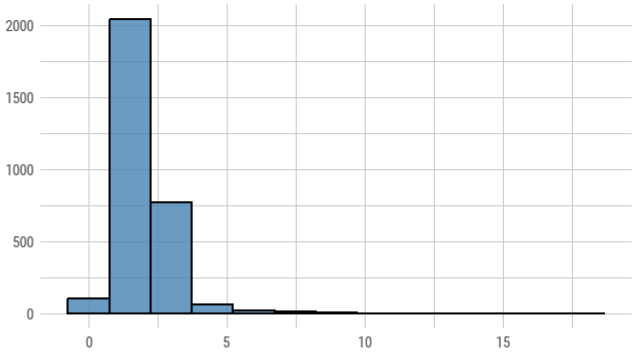


Outlier Diagnosis Plot (test_X2)

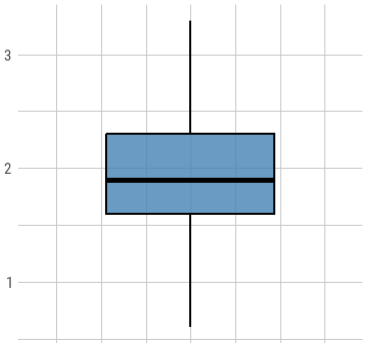
With outliers



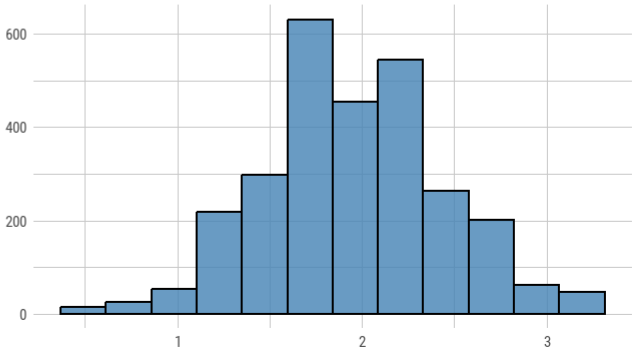
With outliers



Without outliers

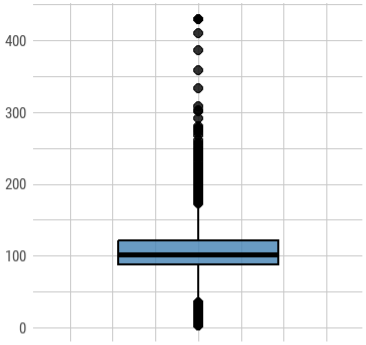


Without outliers

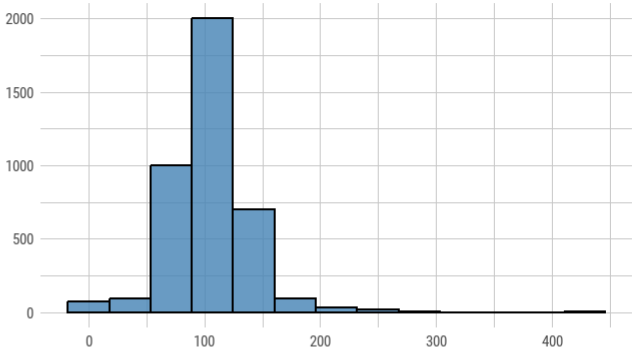


Outlier Diagnosis Plot (test_X3)

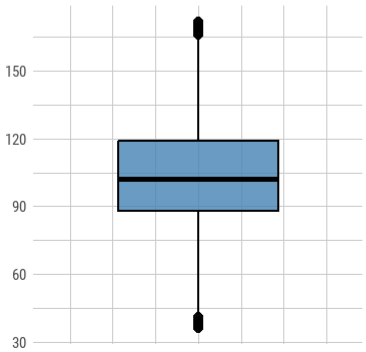
With outliers



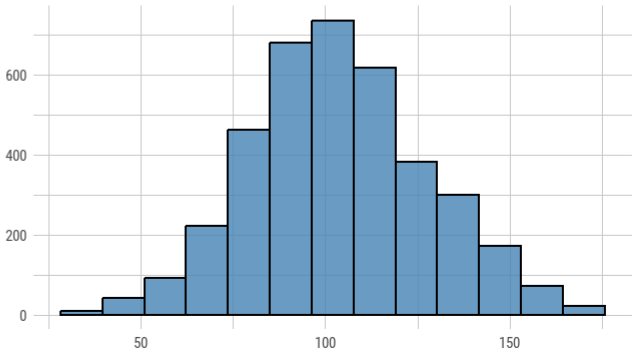
With outliers



Without outliers

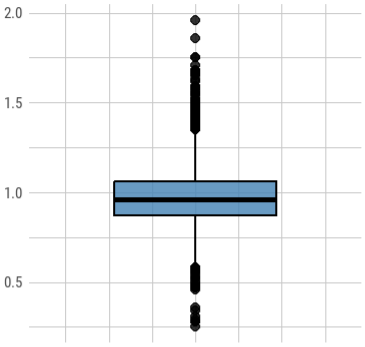


Without outliers

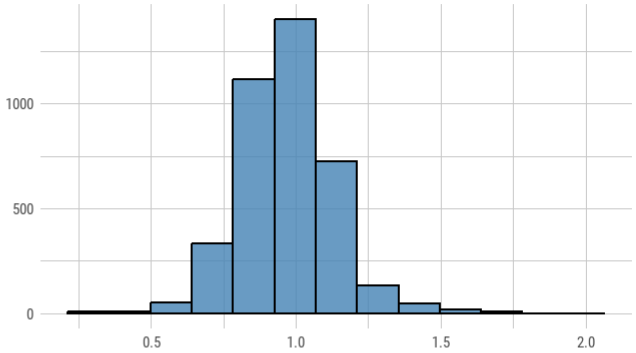


Outlier Diagnosis Plot (test_X4)

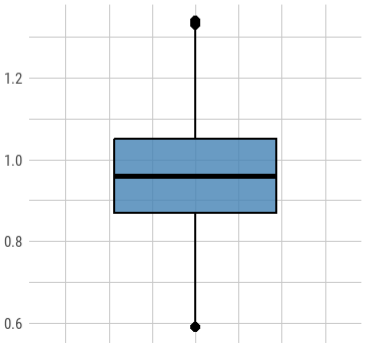
With outliers



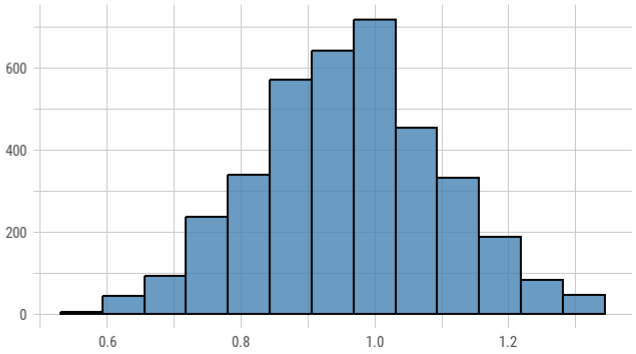
With outliers



Without outliers

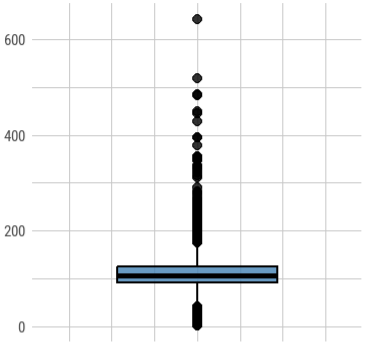


Without outliers

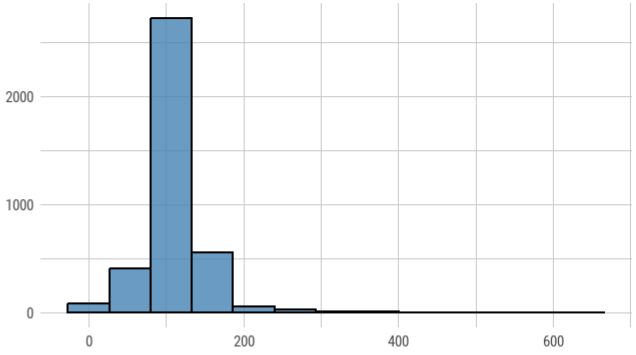


Outlier Diagnosis Plot (test_X5)

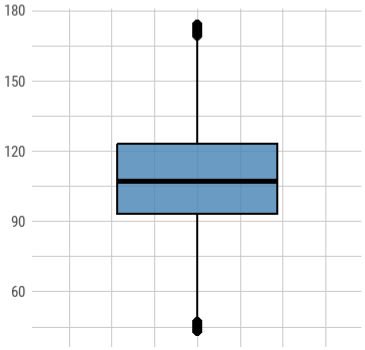
With outliers



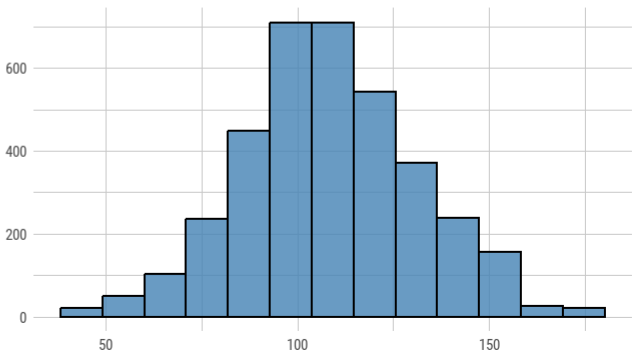
With outliers

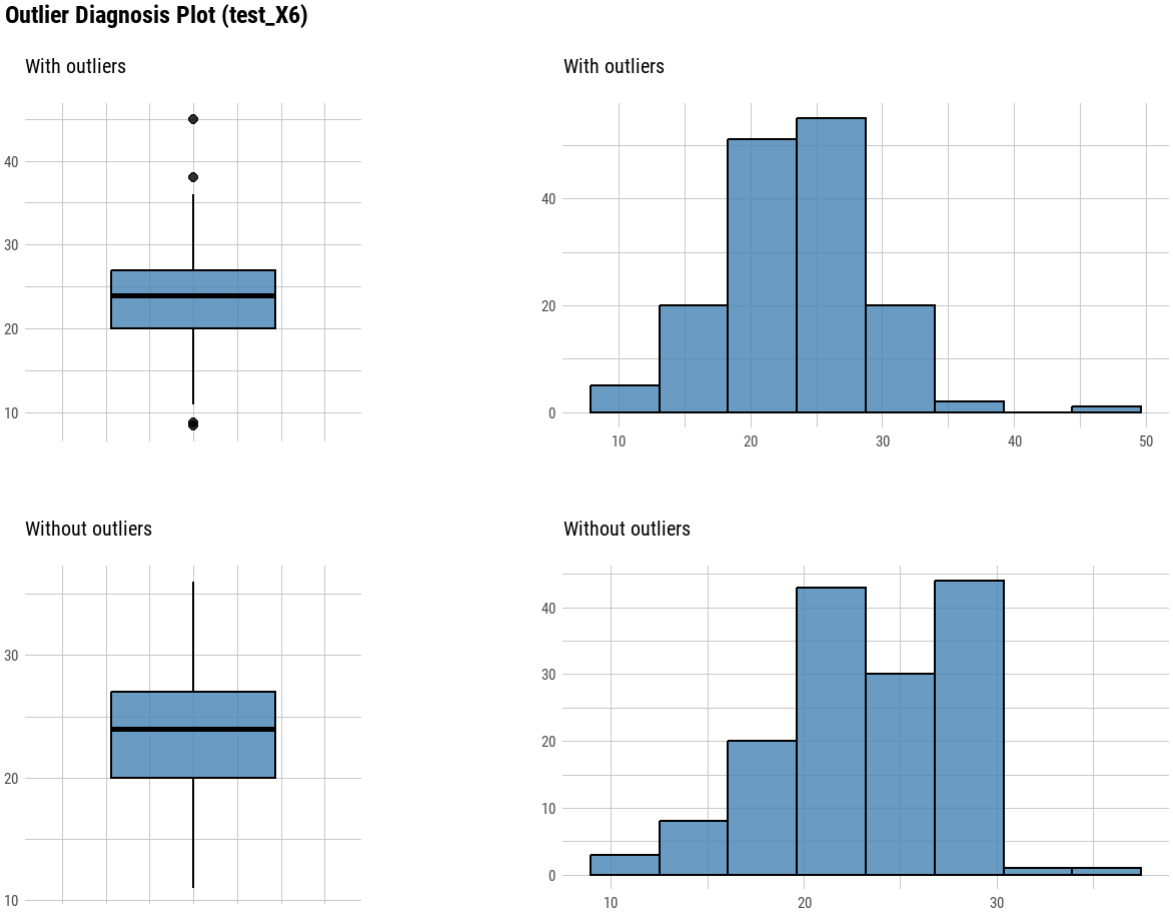


Without outliers



Without outliers





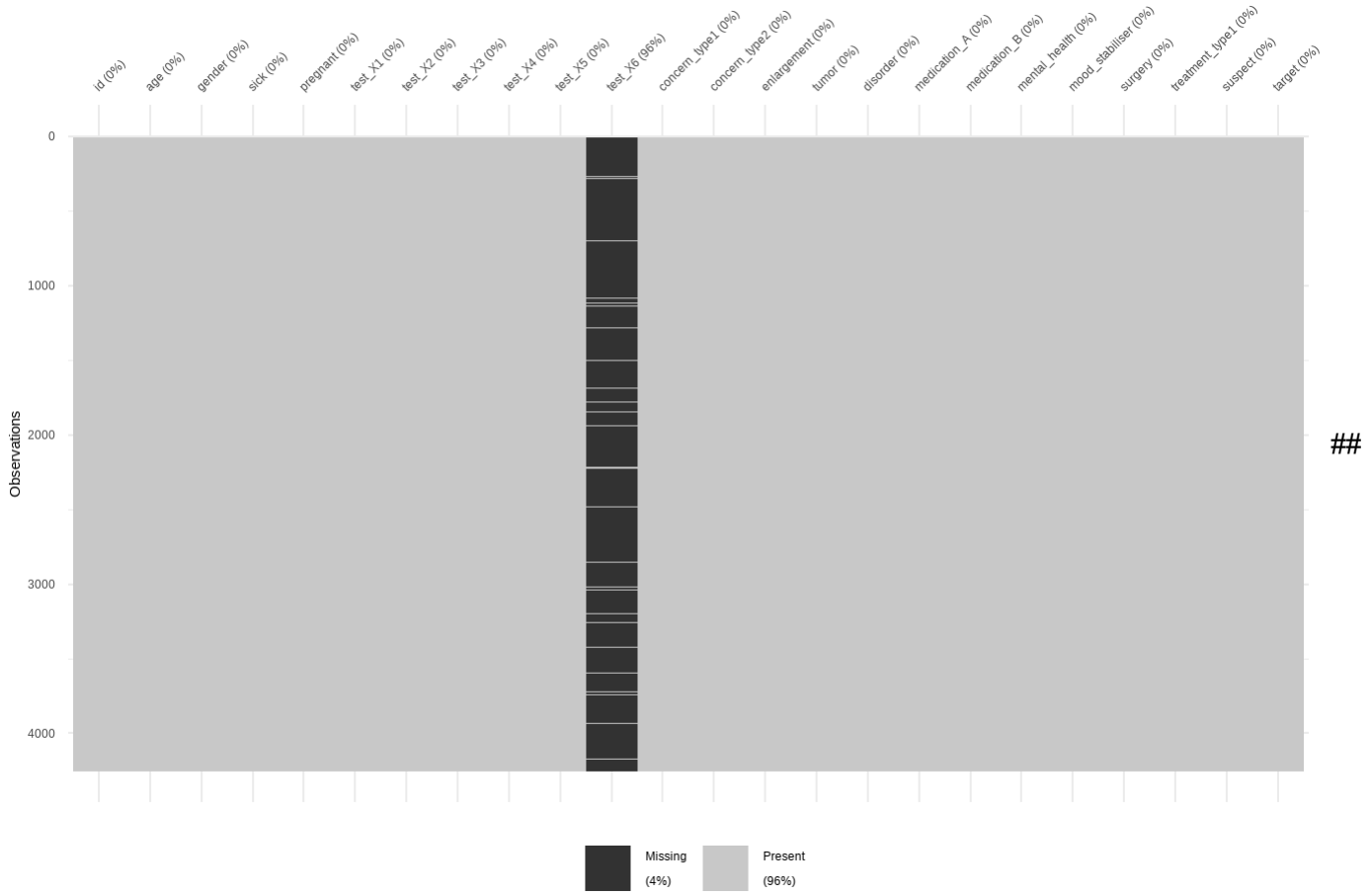
Cleaning the data and handling data issues

Handling missing variables and outliers

```
Train_corrected <- Train %>% mutate(test_X3 = as.numeric( imputate_na(Train, xvar =test_X3, meth
od = "mean")) %>% as.data.frame(),
                                    test_X2 = as.numeric( imputate_na(Train, xvar =test_X2, meth
od = "mean")) %>% as.data.frame(),
                                    test_X1 = as.numeric( imputate_na(Train, xvar =test_X1, meth
od = "mean")) %>% as.data.frame(),
                                    test_X4 = as.numeric( imputate_na(Train, xvar =test_X4, meth
od = "mean")) %>% as.data.frame(),
                                    test_X5 = as.numeric( imputate_na(Train, xvar =test_X5, meth
od = "mean")) %>% as.data.frame() )
library(visdat)
```

```
## Warning: package 'visdat' was built under R version 4.3.3
```

```
vis_miss(Train_corrected)
```



##

Checking if there is duplicate

```
length(unique(Train_corrected$id))
```

```
## [1] 4250
```

```
length(Train_corrected$id)
```

```
## [1] 4250
```

We remove test_X6 due to the percent of missing values present in it (96%)

```
Train_corrected <-Train_corrected %>% select(-c(test_X6, id))
```

Checking the percentage of empty rows to the decide

if to remove it or not

```
n <- Train_corrected %>% filter(gender == "" )
a <-length(n$sick)
s <- length(Train_corrected$sick)
(a/s)*100
```

```
## [1] 3.317647
```

Filtering out the rows with empty space

```
Train_corrected <- Train_corrected %>% filter(!gender =="")
```

Data exploration

Descriptive Analysis

```
library(summarytools)
```

```
## Warning: package 'summarytools' was built under R version 4.3.3
```

```
##
## Attaching package: 'summarytools'
```

```
## The following object is masked from 'package:tibble':
##
##     view
```

```
K1 <- Train_corrected %>% select(starts_with("test_X"))

summary(K1)
```

```
##      test_X1..           test_X2..          test_X3..           test_X4..
##  Min.   :  0.0050   Min.   : 0.050000   Min.   :  2.0000   Min.   :0.2500000
##  1st Qu.:  0.7000   1st Qu.: 1.700000   1st Qu.: 88.0000   1st Qu.:0.8800000
##  Median :  1.7000   Median : 2.035580   Median :104.0000   Median :0.9708460
##  Mean   :  7.2855   Mean   : 2.029092   Mean   :104.8029   Mean   :0.9717627
##  3rd Qu.:  4.4000   3rd Qu.: 2.100000   3rd Qu.:120.0000   3rd Qu.:1.0500000
##  Max.   :530.0000   Max.   :18.000000   Max.   :430.0000   Max.   :1.9600000
##      test_X5..
##  Min.   :  1.400
##  1st Qu.: 94.000
##  Median :110.000
##  Mean   :109.593
##  3rd Qu.:123.000
##  Max.   :642.000
```

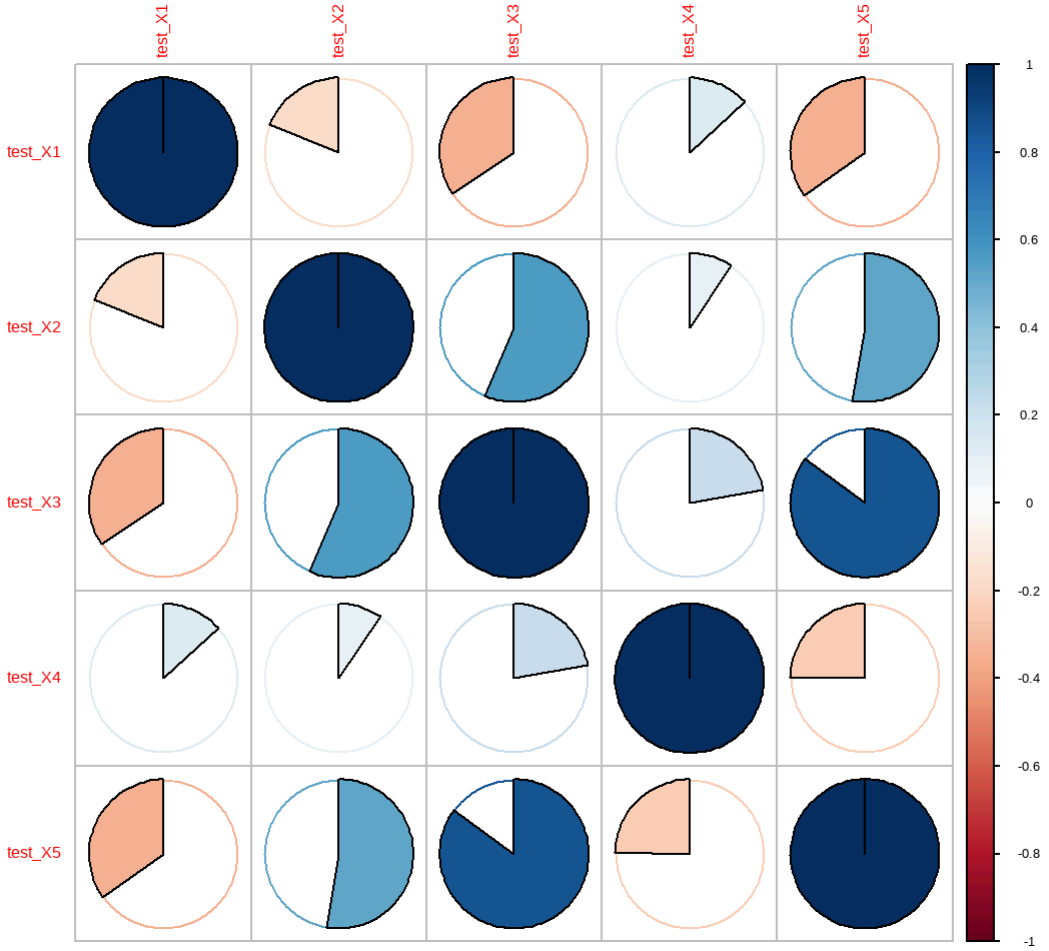
Correlation Analysis

```
library(corrplot)
```

```
## Warning: package 'corrplot' was built under R version 4.3.2
```

```
## corrplot 0.92 loaded
```

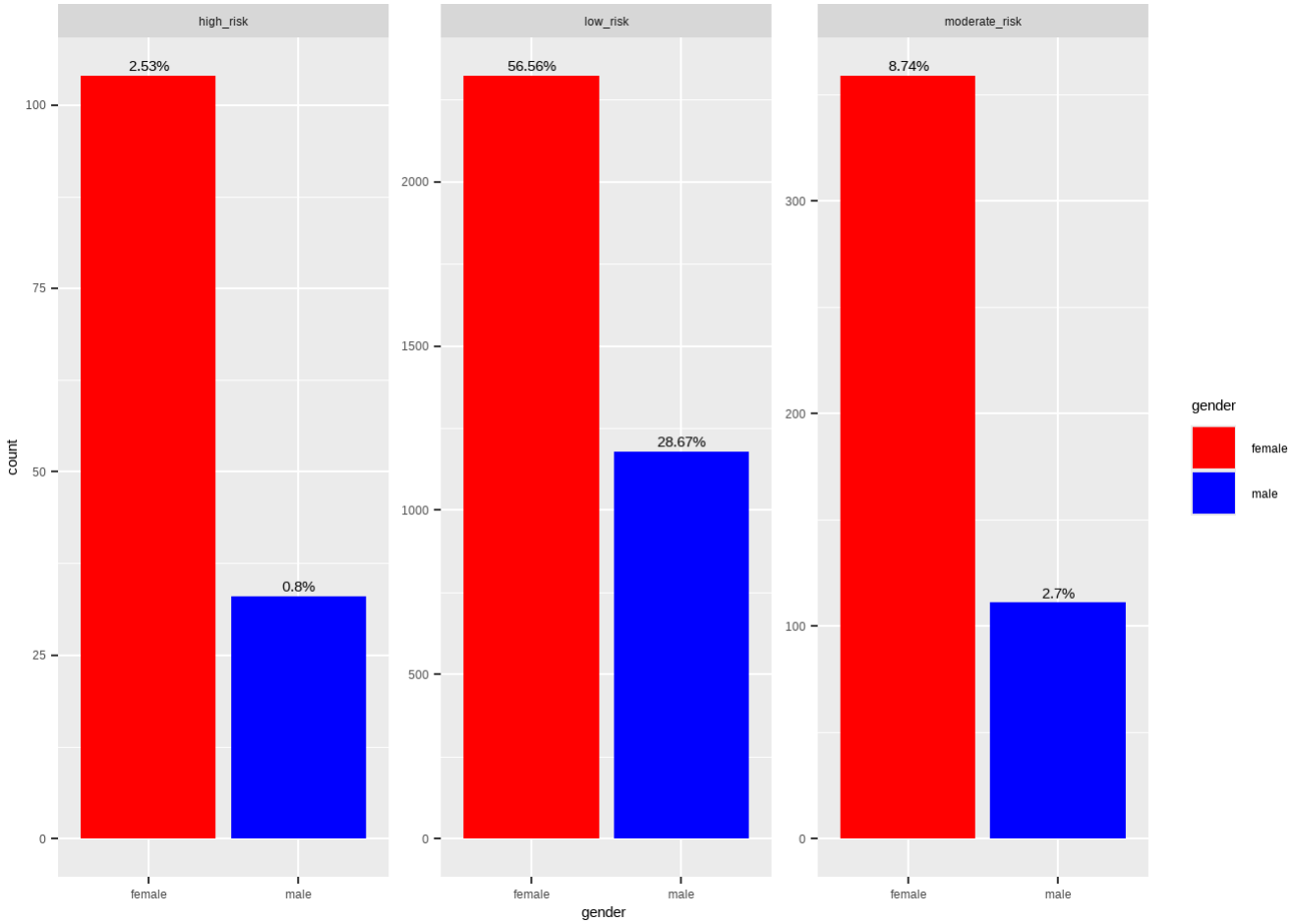
```
l <- cor(K1)
corrplot(l, "pie")
```



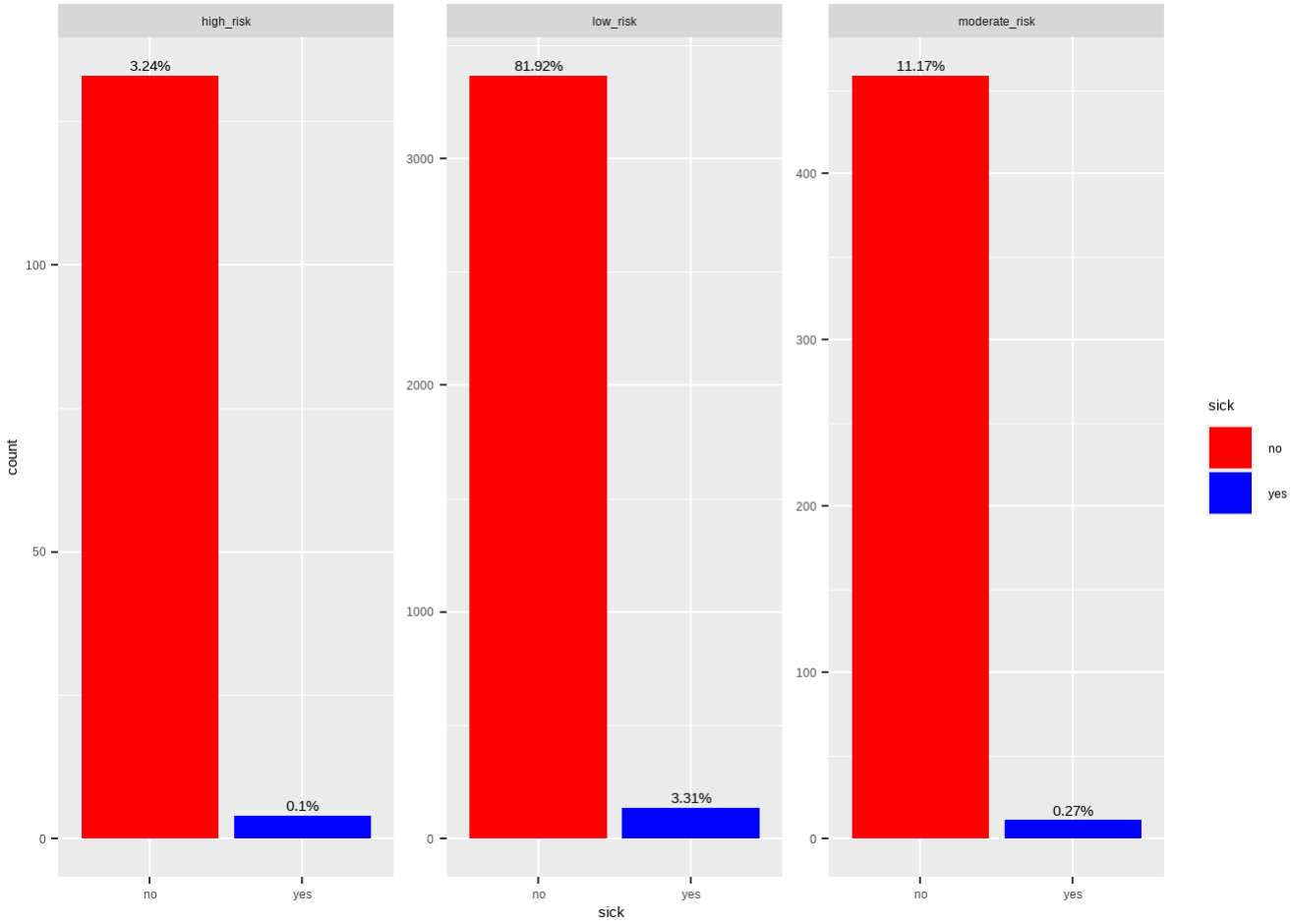
Visualization

```
ggplot(Train_corrected, aes(x = gender, fill = gender)) +
  geom_bar() +
  geom_text(stat = "count", aes(label = paste0(round((..count..)/sum(..count..)*100, 2), "%")),
vjust = -0.5) +
  facet_wrap(~ target, scale = "free") +
  scale_fill_manual(values = c("red", "blue"))
```

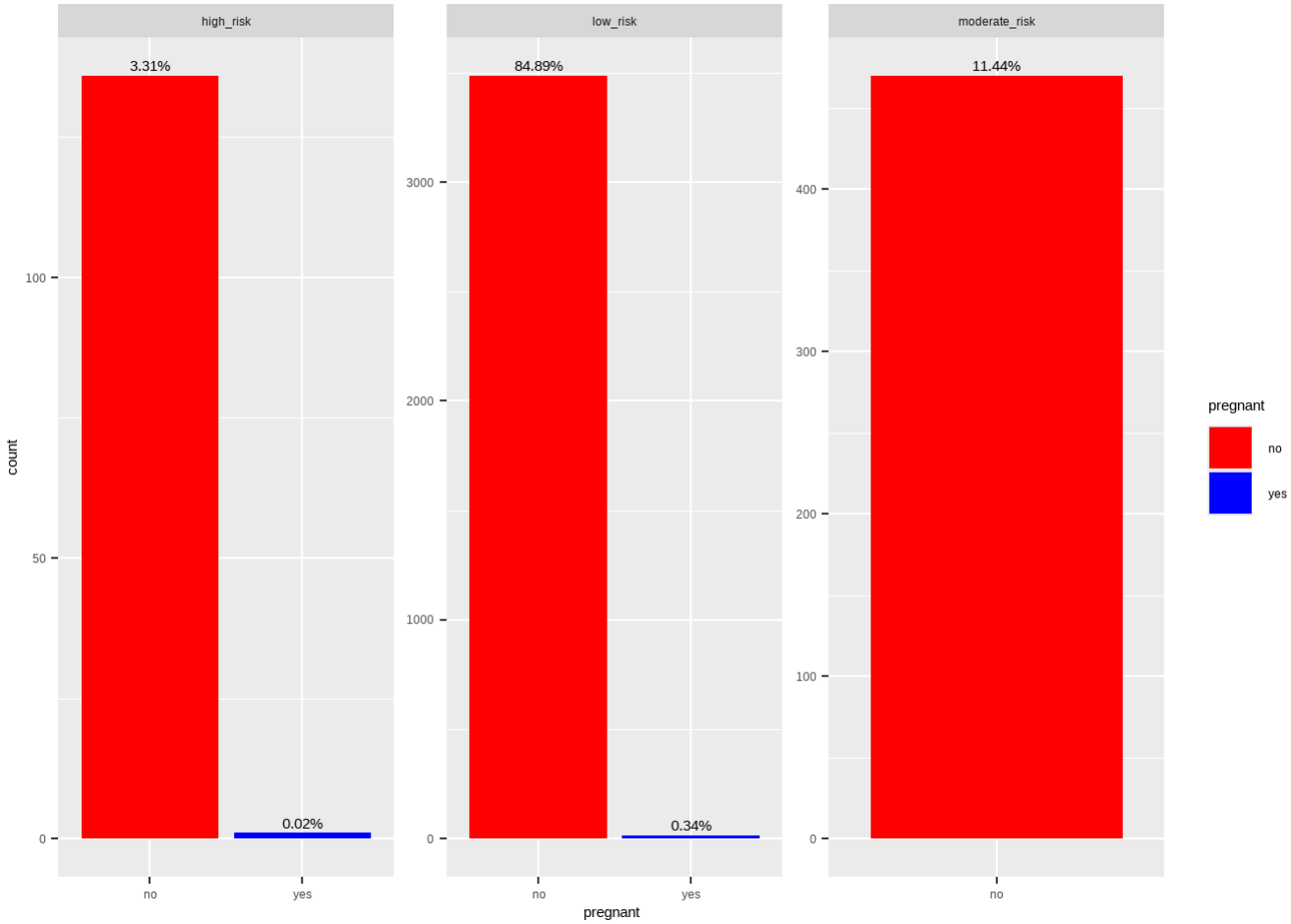
```
## Warning: The dot-dot notation (`..count..`) was deprecated in ggplot2 3.4.0.
## ℹ Please use `after_stat(count)` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```



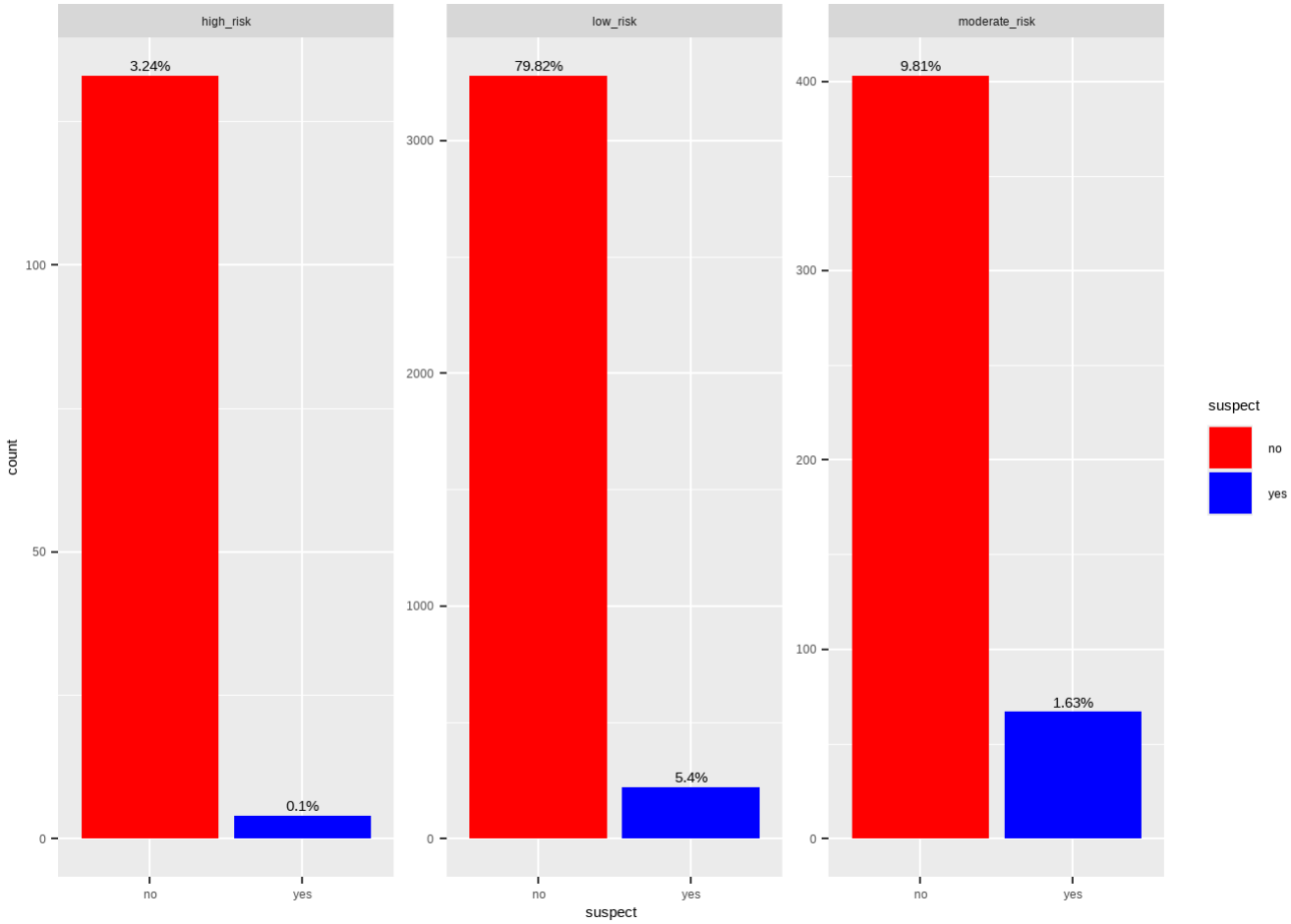

```
ggplot(Train_corrected, aes(x=sick, fill = sick))+geom_bar() + geom_text(stat = "count", aes(lab
el = paste0(round((..count..)/sum(..count..)*100, 2), "%")), vjust = -0.5) +
  facet_wrap(~ target, scale = "free") +
  scale_fill_manual(values = c("red", "blue"))
```



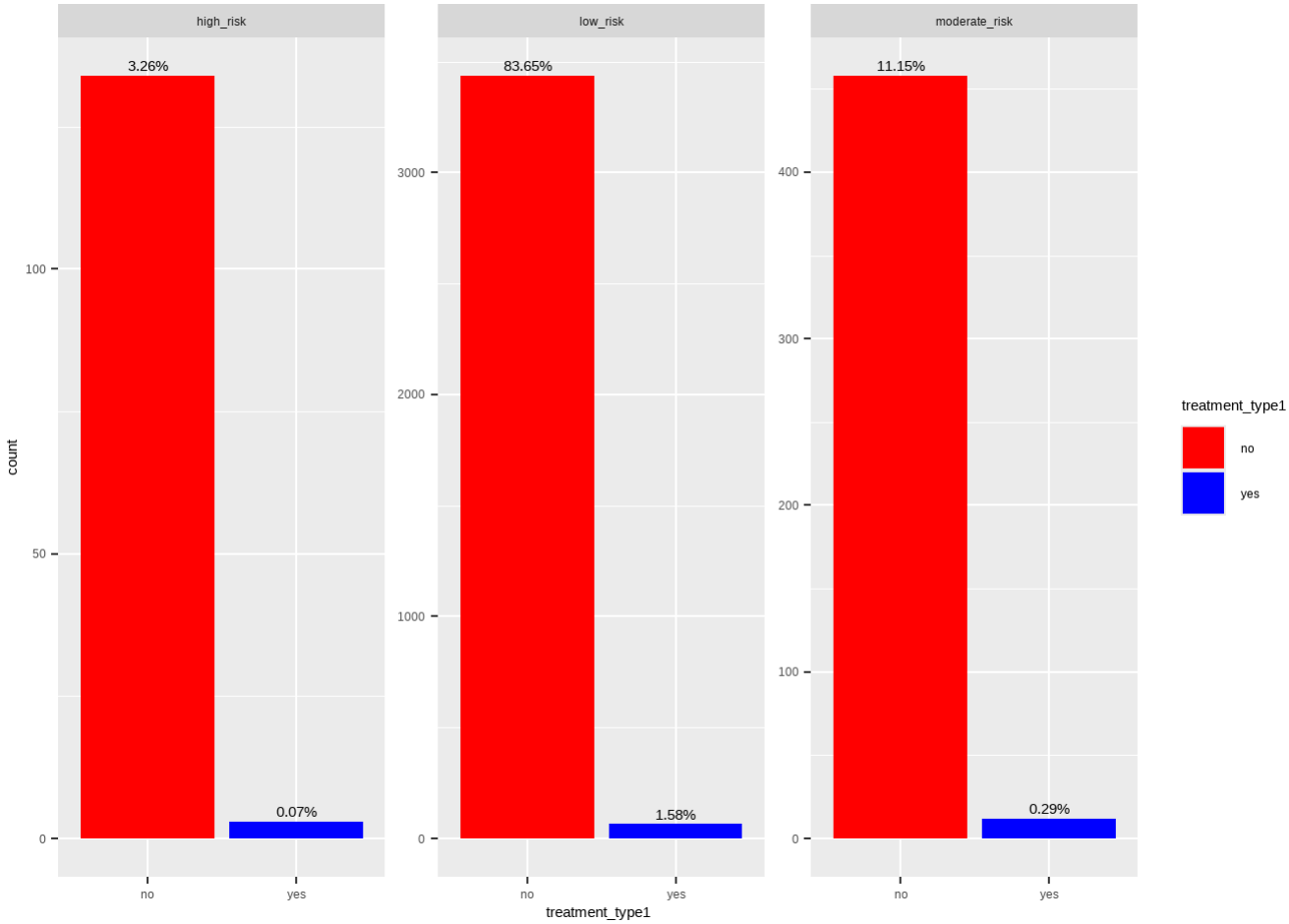

```
ggplot(Train_corrected, aes(x=pregnant, fill= pregnant))+geom_bar() + geom_text(stat = "count",
aes(label = paste0(round((..count..)/sum(..count..)*100, 2), "%")), vjust = -0.5) +
  facet_wrap(~ target, scale = "free") +
  scale_fill_manual(values = c("red", "blue"))
```



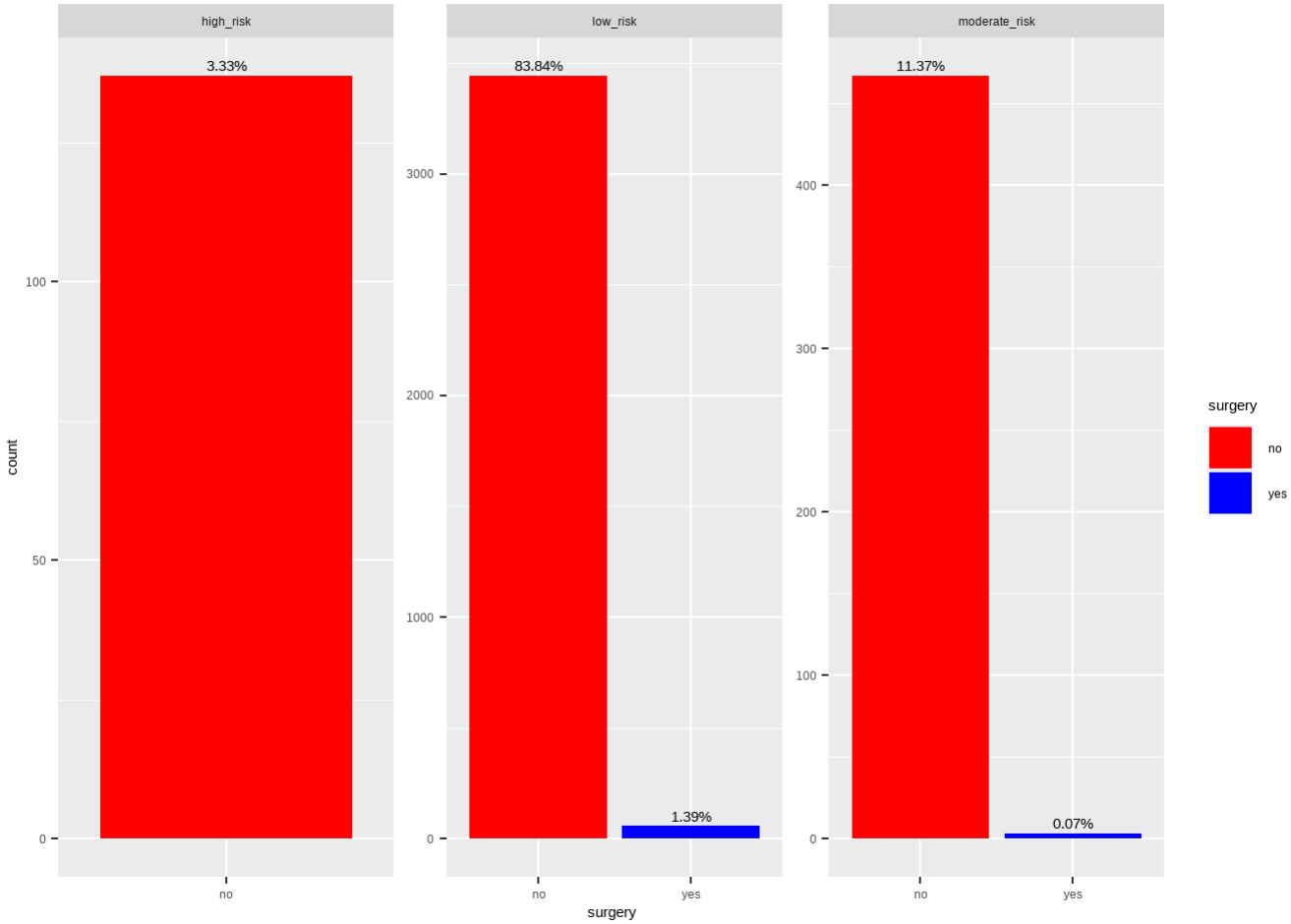

```
ggplot(Train_corrected, aes(x= suspect, fill=suspect))+geom_bar() + geom_text(stat = "count", aes(label = paste0(round((..count..)/sum(..count..)*100, 2), "%")), vjust = -0.5) +
  facet_wrap(~ target, scale = "free") +
  scale_fill_manual(values = c("red", "blue"))
```



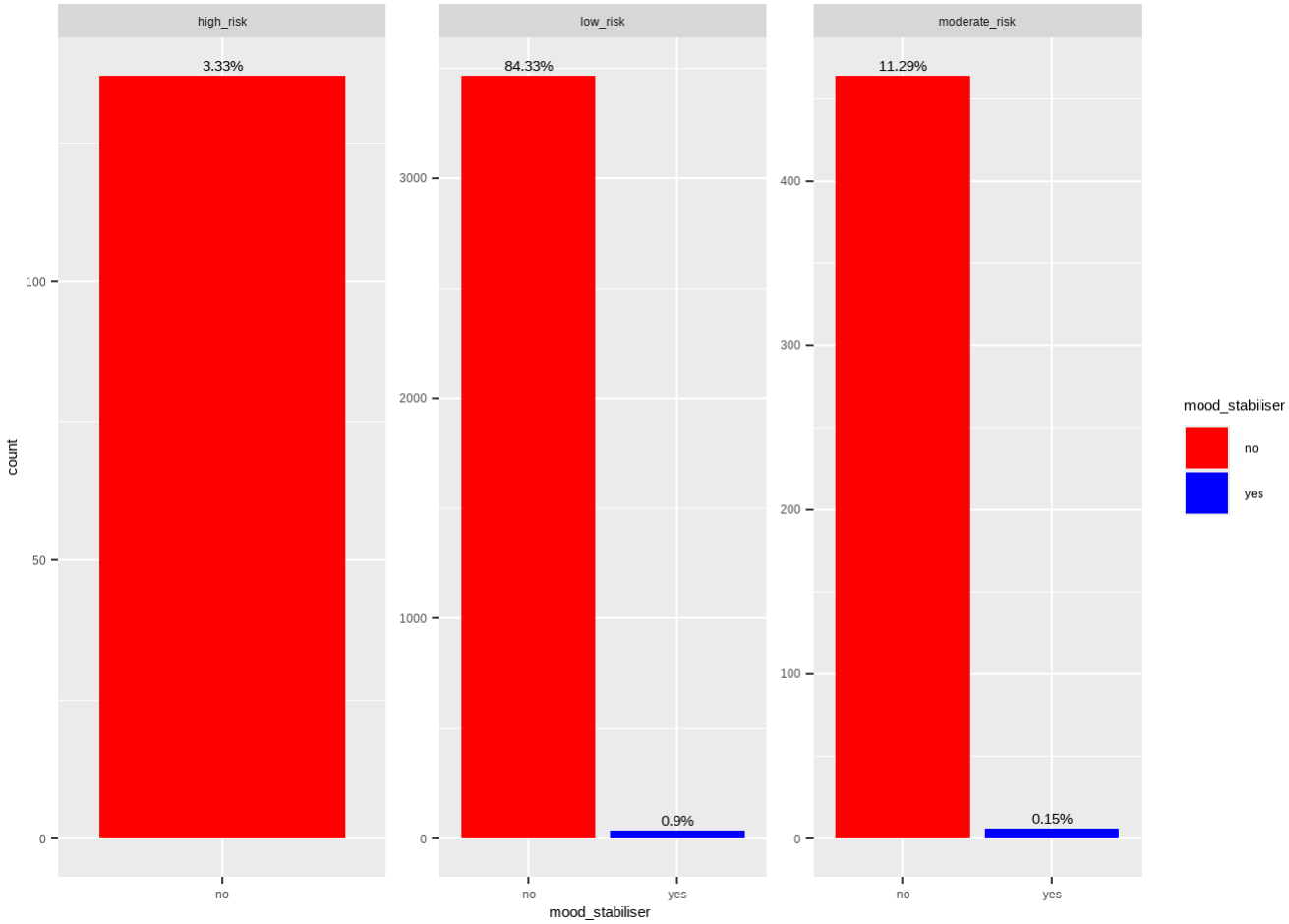

```
ggplot(Train_corrected, aes(x=treatment_type1, fill=treatment_type1))+geom_bar() + geom_text(sta
t = "count", aes(label = paste0(round((..count..)/sum(..count..)*100, 2), "%")), vjust = -0.5) +
  facet_wrap(~ target, scale = "free") +
  scale_fill_manual(values = c("red", "blue"))
```



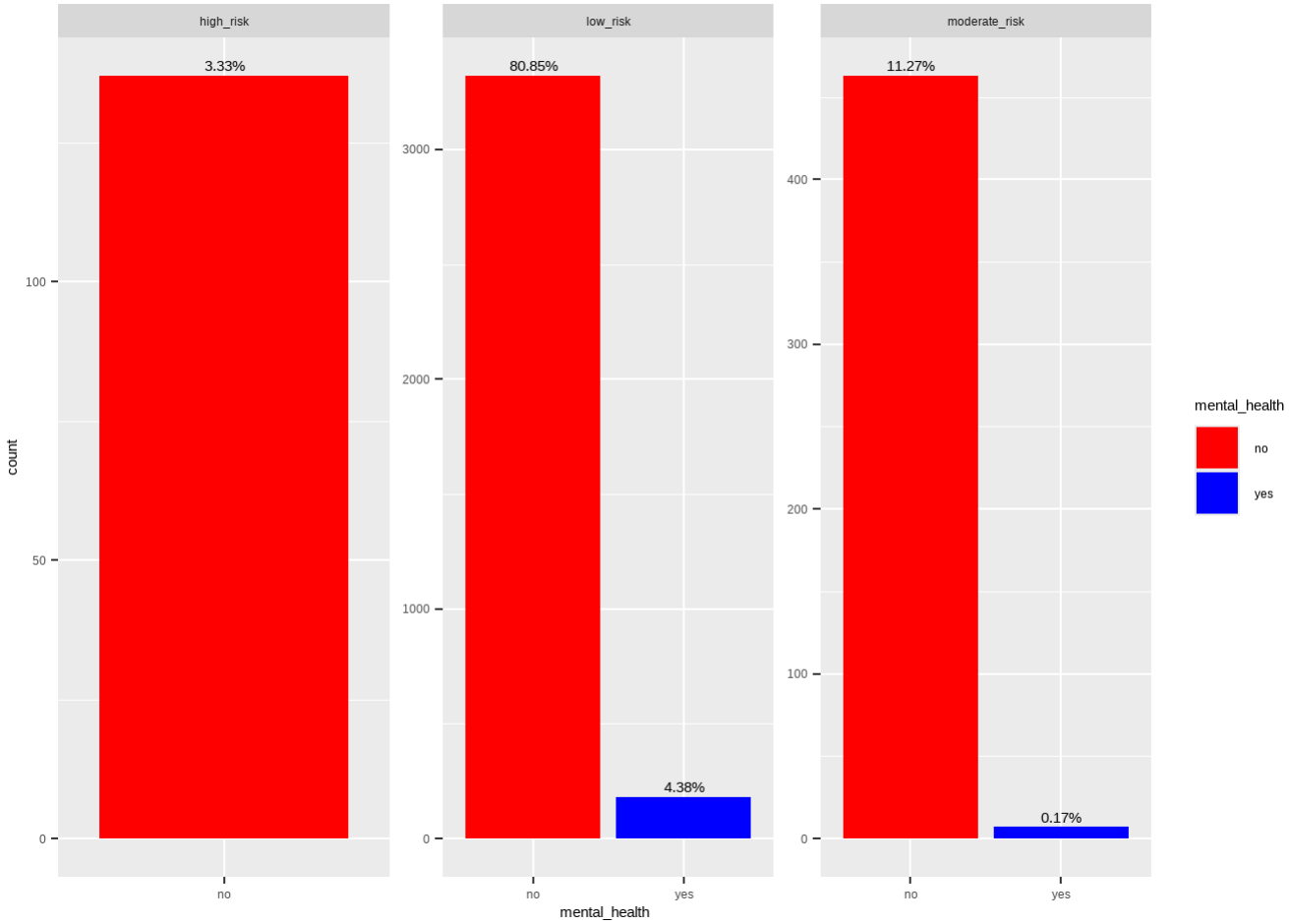

```
ggplot(Train_corrected, aes(x=surgery, fill= surgery))+geom_bar() + geom_text(stat = "count", aes(label = paste0(round((..count..)/sum(..count..)*100, 2), "%")), vjust = -0.5) +
  facet_wrap(~ target, scale = "free") +
  scale_fill_manual(values = c("red", "blue"))
```



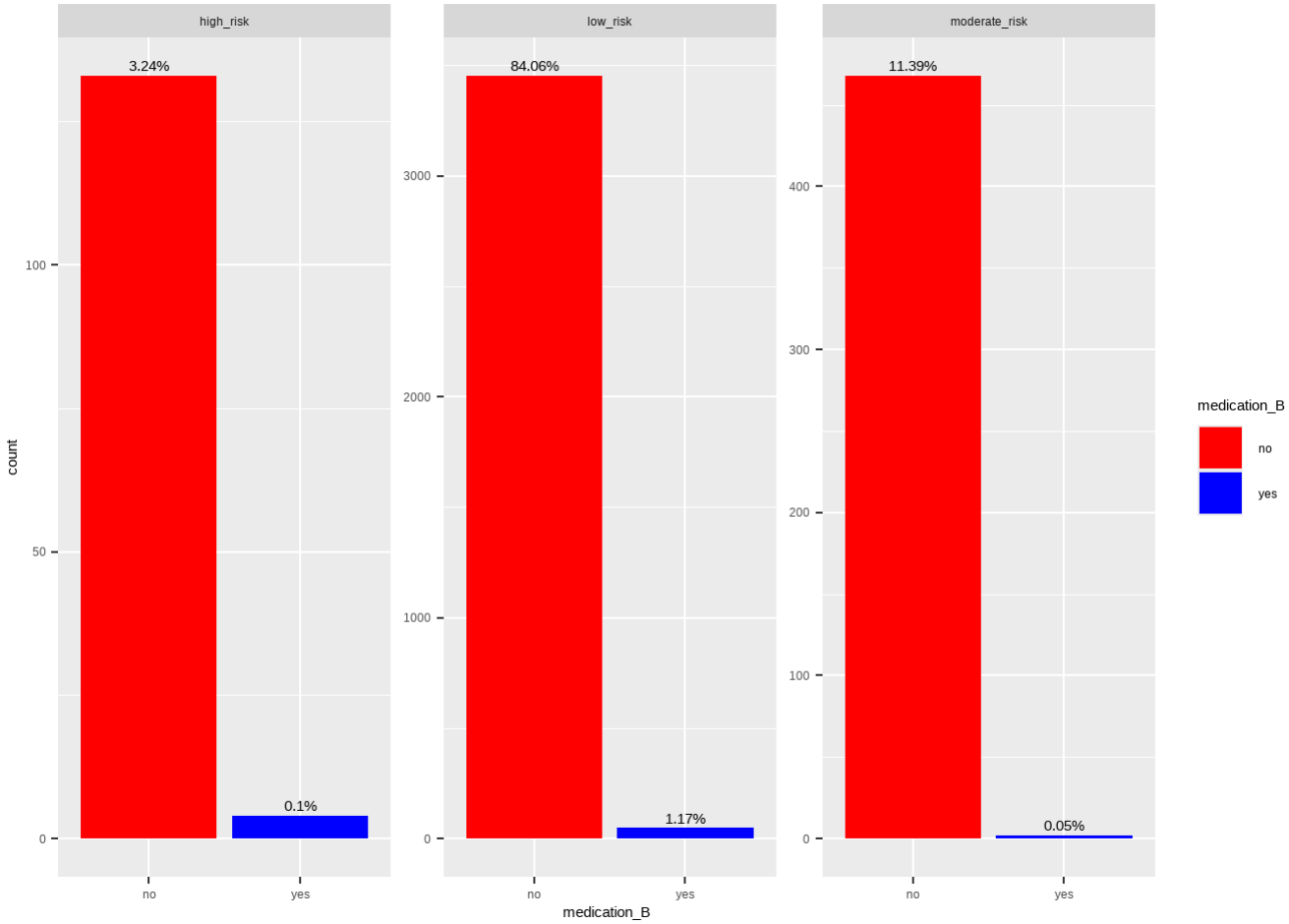

```
ggplot(Train_corrected, aes(x=mood_stabiliser, fill= mood_stabiliser))+geom_bar() + geom_text(st
at = "count", aes(label = paste0(round((..count..)/sum(..count..)*100, 2), "%")), vjust = -0.5)
+
  facet_wrap(~ target, scale = "free") +
  scale_fill_manual(values = c("red", "blue"))
```



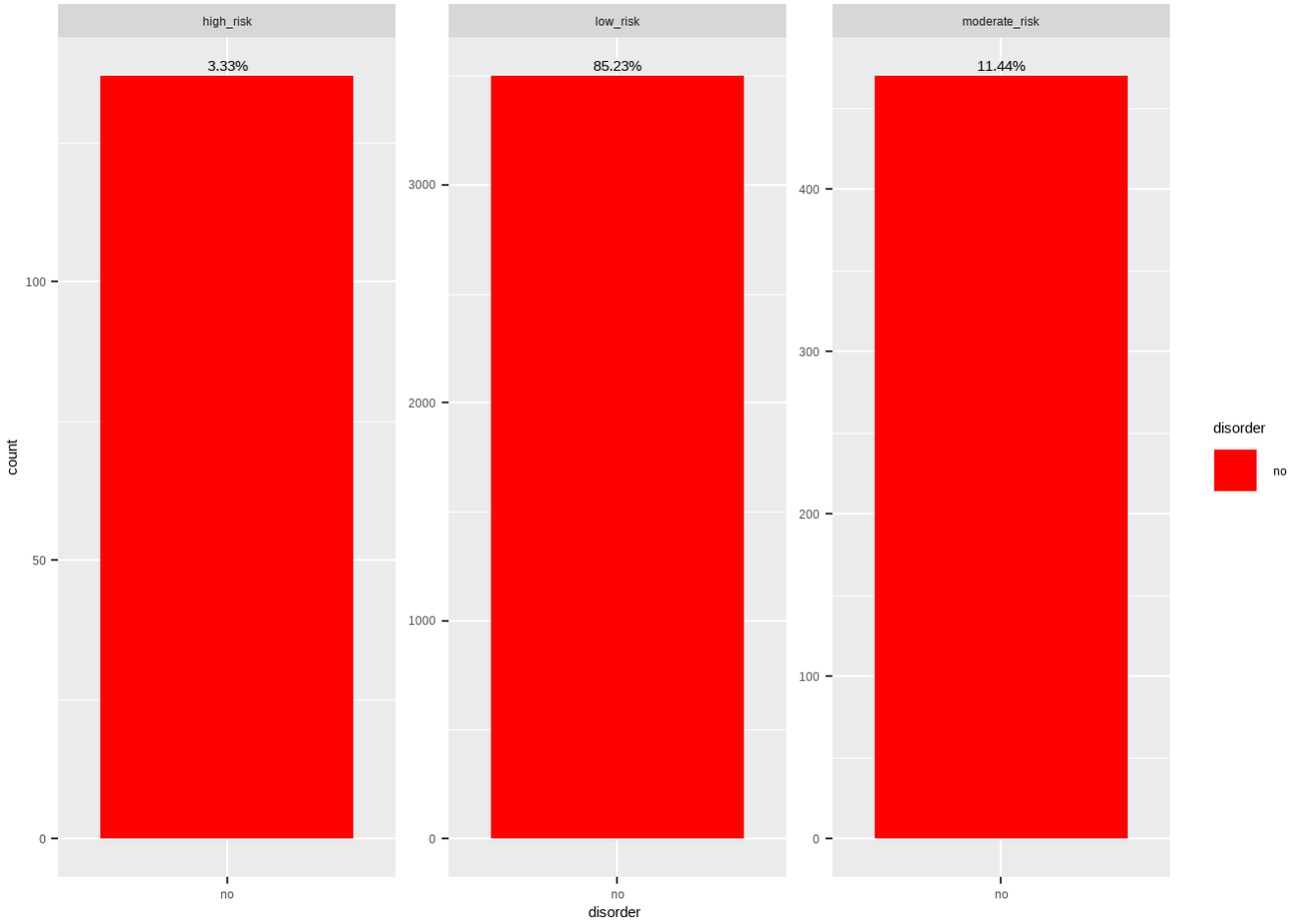

```
ggplot(Train_corrected, aes(x=mental_health, fill=mental_health))+geom_bar() + geom_text(stat =
"count", aes(label = paste0(round((..count..)/sum(..count..)*100, 2), "%")), vjust = -0.5) +
  facet_wrap(~ target, scale = "free") +
  scale_fill_manual(values = c("red", "blue"))
```



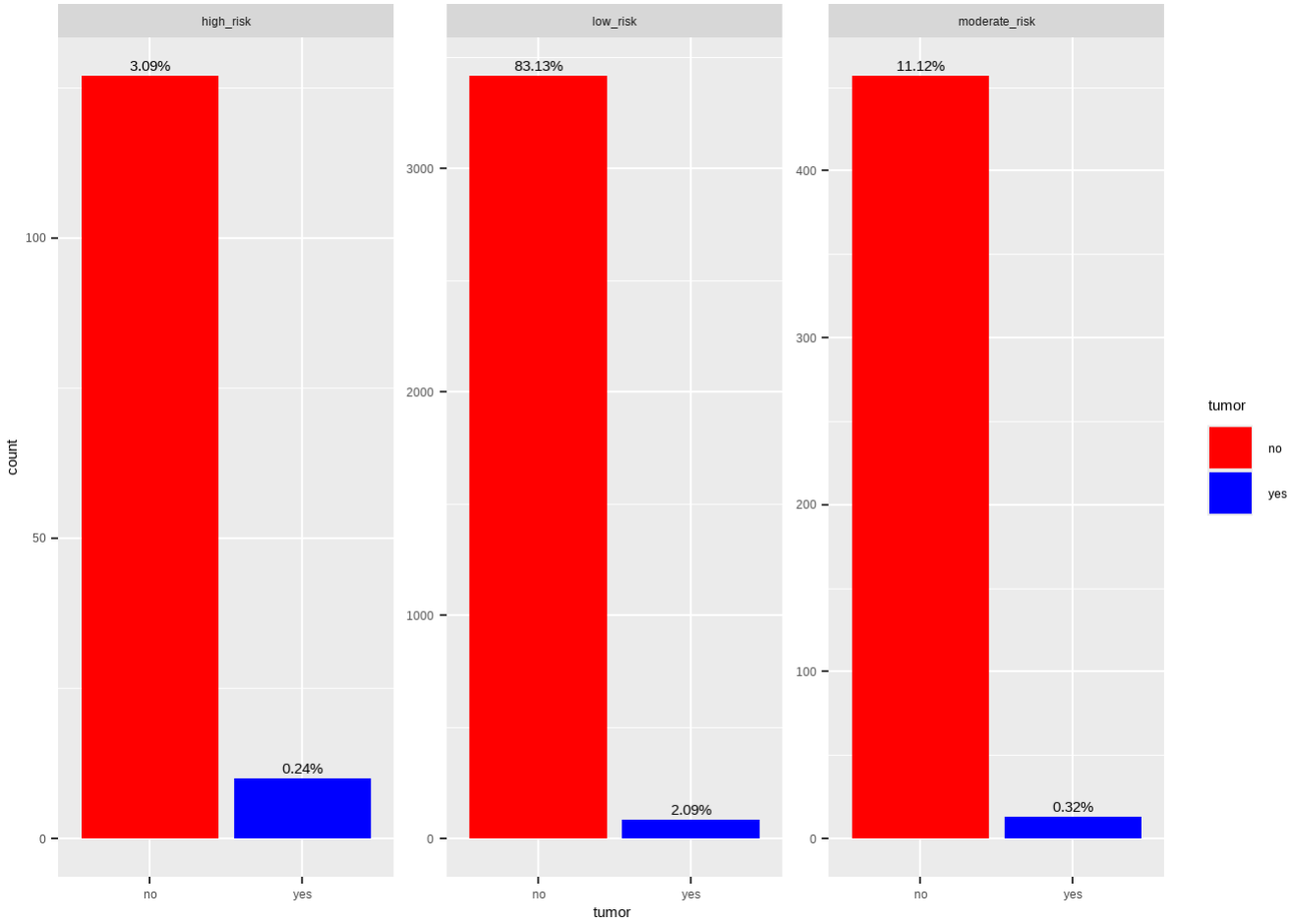

```
ggplot(Train_corrected, aes(x=medication_B, fill=medication_B))+geom_bar() + geom_text(stat = "c
ount", aes(label = paste0(round((..count..)/sum(..count..)*100, 2), "%")), vjust = -0.5) +
  facet_wrap(~ target, scale = "free") +
  scale_fill_manual(values = c("red", "blue"))
```



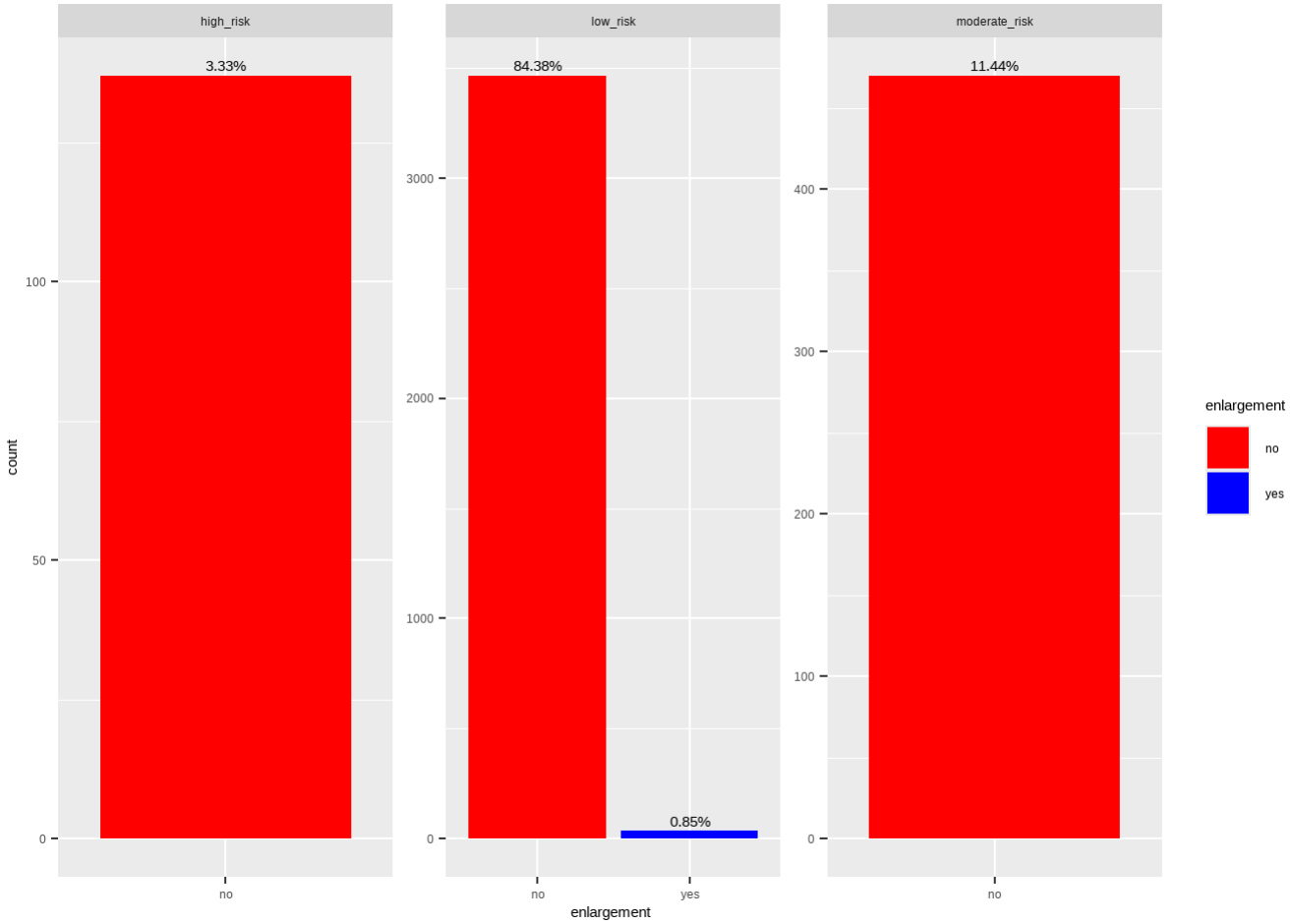

```
ggplot(Train_corrected, aes(x=disorder, fill=disorder))+geom_bar() + geom_text(stat = "count", a
es(label = paste0(round((..count..)/sum(..count..)*100, 2), "%")), vjust = -0.5) +
  facet_wrap(~ target, scale = "free") +
  scale_fill_manual(values = c("red", "blue"))
```



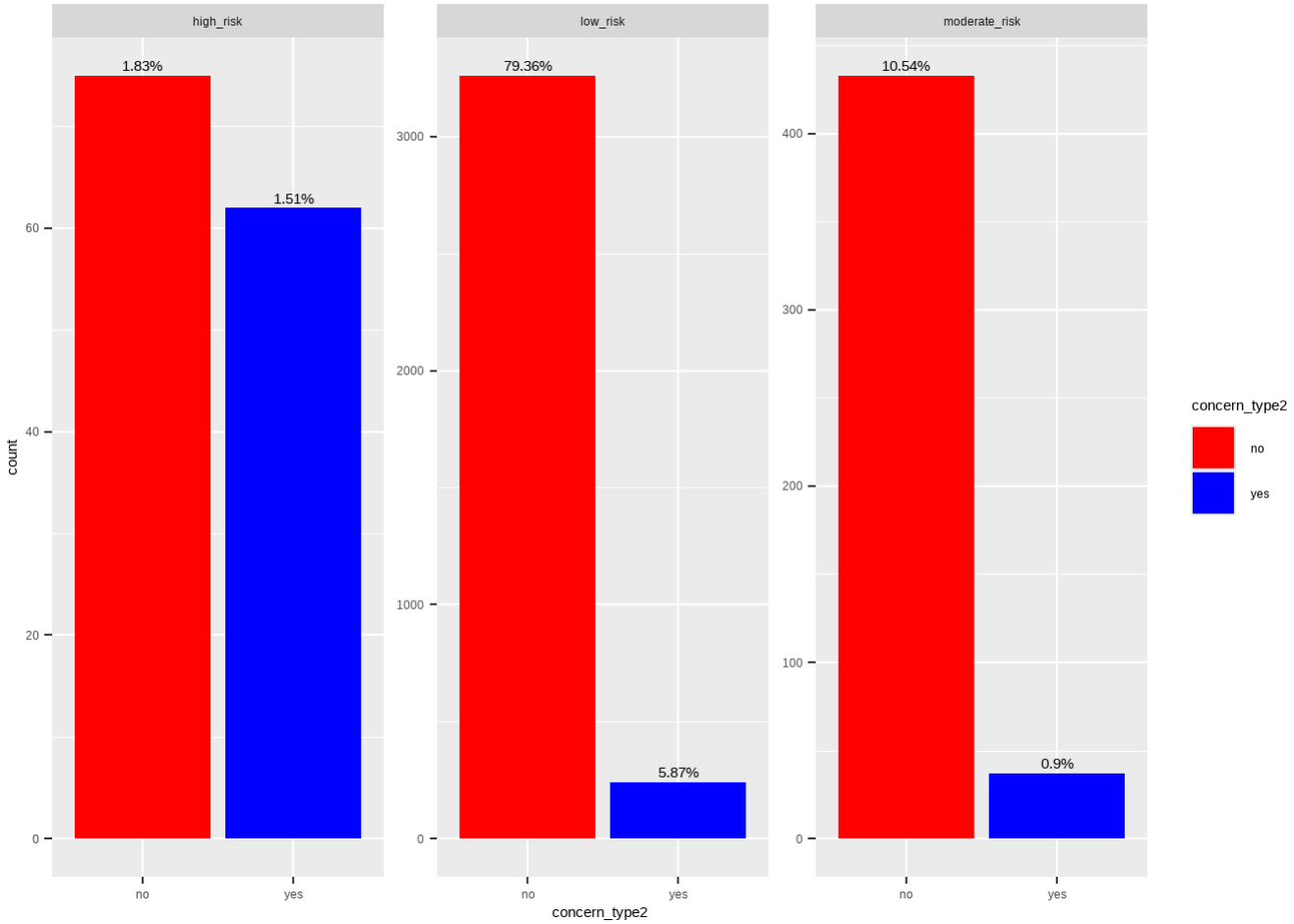

```
ggplot(Train_corrected, aes(x=tumor, fill=tumor))+geom_bar() + geom_text(stat = "count", aes(lab
el = paste0(round((..count..)/sum(..count..)*100, 2), "%")), vjust = -0.5) +
  facet_wrap(~ target, scale = "free") +
  scale_fill_manual(values = c("red", "blue"))
```



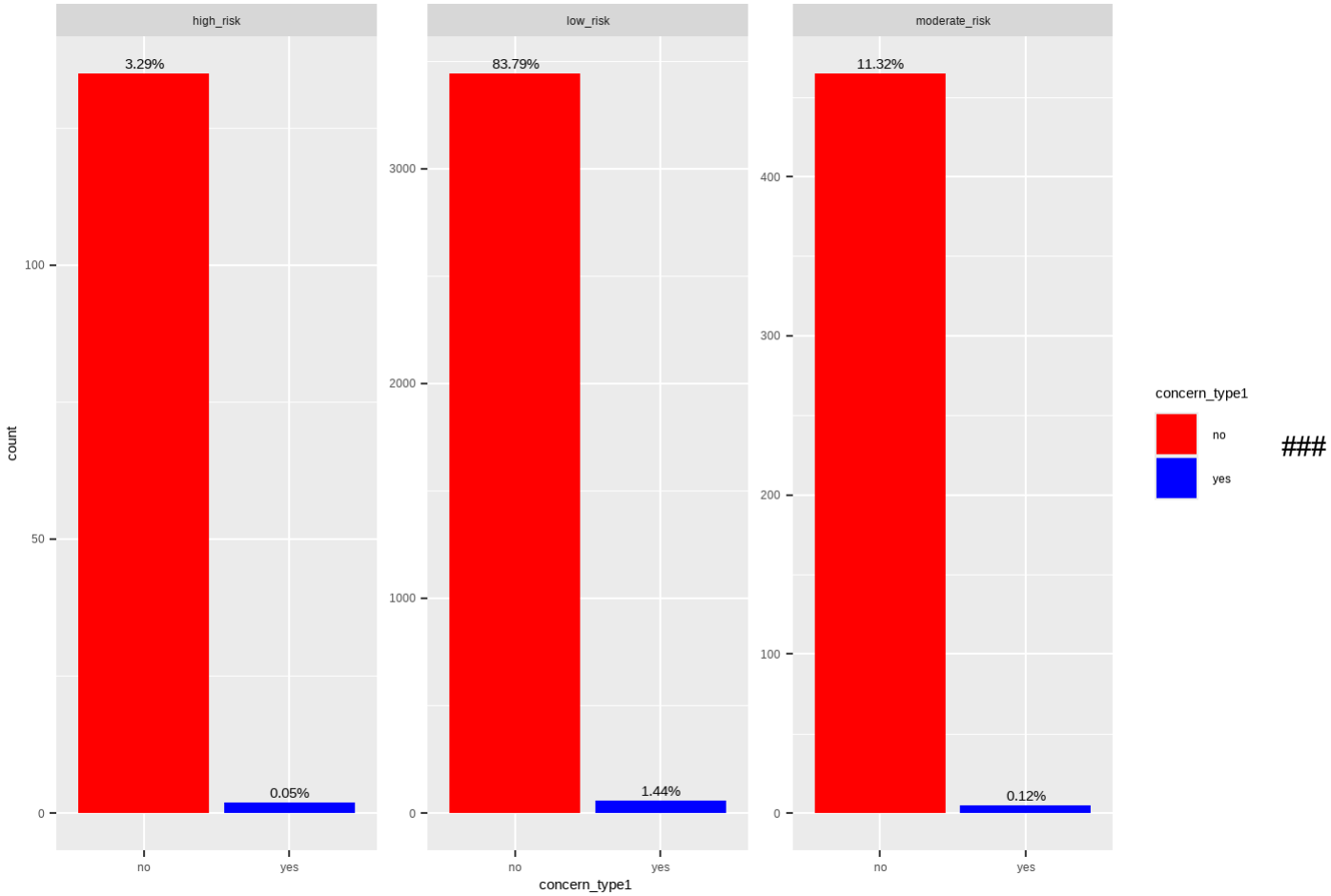

```
ggplot(Train_corrected, aes(x=enlargement, fill=enlargement))+geom_bar() + geom_text(stat = "cou
nt", aes(label = paste0(round((..count..)/sum(..count..)*100, 2), "%")), vjust = -0.5) +
  facet_wrap(~ target, scale = "free") +
  scale_fill_manual(values = c("red", "blue"))
```




```
ggplot(Train_corrected, aes(x=concern_type2, fill= concern_type2))+geom_bar() + geom_text(stat =
"count", aes(label = paste0(round((..count..)/sum(..count..)*100, 2), "%")), vjust = -0.5) +
  facet_wrap(~ target, scale = "free") +
  scale_fill_manual(values = c("red", "blue"))
```




```
ggplot(Train_corrected, aes(x=concern_type1, fill=concern_type1))+geom_bar() + geom_text(stat =
"count", aes(label = paste0(round((..count..)/sum(..count..)*100, 2), "%")), vjust = -0.5) +
  facet_wrap(~ target, scale = "free") +
  scale_fill_manual(values = c("red", "blue"))
```



Data preprocessing - Scaling method

```
Model_data <- scale(K1)
```

Adding the Target variable to model data

```
target <- Train_corrected %>% select(target)
Model_data <- cbind(Model_data, target)
```

Splitting the dataset into 25% test and 75% train set

```
library(caTools)
sample <- sample.split(Model_data$target, SplitRatio = 0.75)
Model_data_Train <- subset(Model_data, sample==T)
Model_data_Test <- subset(Model_data, sample==F)
```

Model buiding and training with parameter tuning - Cart Decision tree

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.3.3
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```
ctrl <- trainControl(method = "cv", number = 10)
tunegrid <- expand.grid(.cp = seq(0.01, 0.5, by = 0.01))
model <- train(target ~ ., data = Model_data, method = "rpart", trControl = ctrl, tuneGrid = tun
egrid)
```

Selecting the best decison tree based on the parameter

```
best_model <- model$finalModel
```

Visualizing the tree

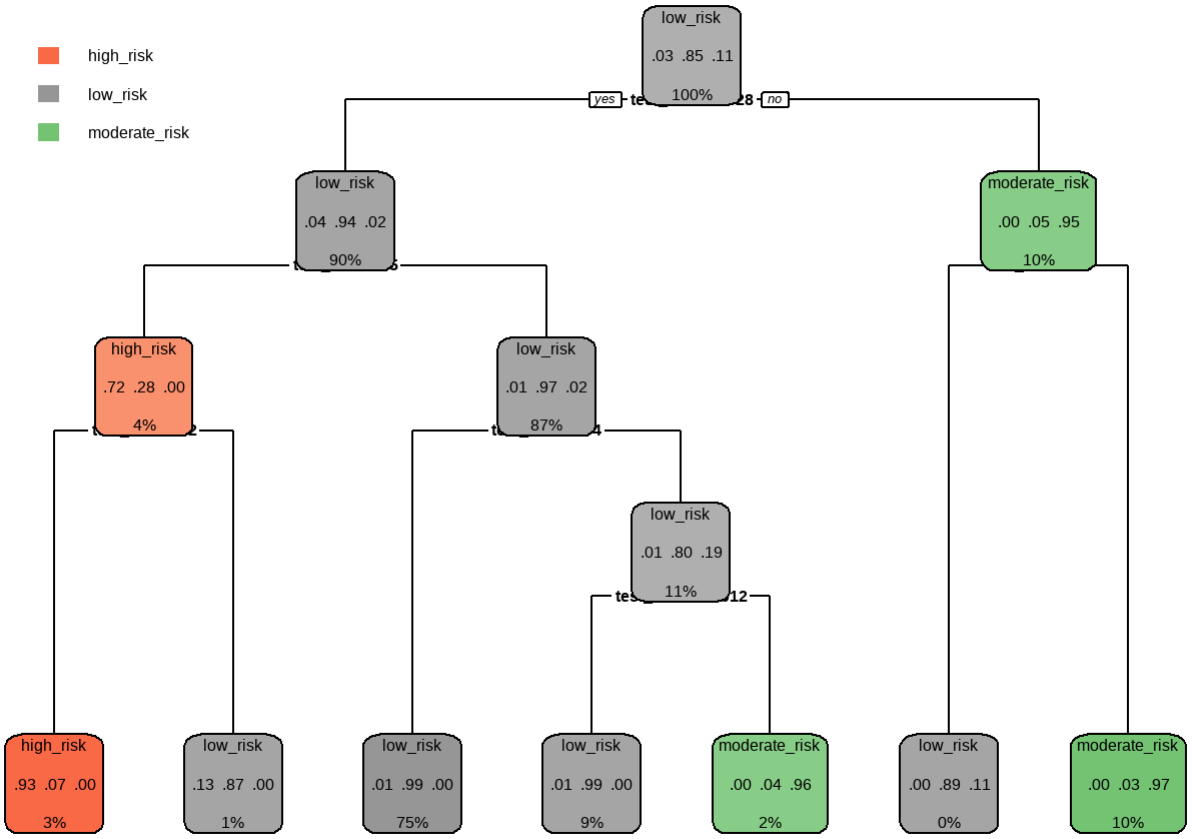
```
library(rpart.plot)
```

```
## Warning: package 'rpart.plot' was built under R version 4.3.3
```

```
## Loading required package: rpart
```

```
## Warning: package 'rpart' was built under R version 4.3.3
```

```
rpart.plot(best_model)
```



Evaluating the performance of the decision tree

```
pp <- predict(best_model, newdata = Model_data_Test, type = "class")
con_pp <- confusionMatrix(as.factor(Model_data_Test$target), pp)
con_pp
```

```
## Confusion Matrix and Statistics
##
##                Reference
## Prediction      high_risk low_risk moderate_risk
##   high_risk            25        9             0
##   low_risk              3      867             6
##   moderate_risk         0        1           117
##
## Overall Statistics
##
##                Accuracy : 0.9815
##                  95% CI : (0.9713, 0.9888)
##     No Information Rate : 0.8531
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9285
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: high_risk Class: low_risk Class: moderate_risk
## Sensitivity                   0.89286          0.9886                0.9512
## Specificity                   0.99100          0.9404                0.9989
## Pos Pred Value                0.73529          0.9897                0.9915
## Neg Pred Value                0.99698          0.9342                0.9934
## Prevalence                    0.02724          0.8531                0.1196
## Detection Rate                0.02432          0.8434                0.1138
## Detection Prevalence          0.03307          0.8521                0.1148
## Balanced Accuracy             0.94193          0.9645                0.9751
```

Building a support vector machine model

```
library(e1071)
```

```
##
## Attaching package: 'e1071'
```

```
## The following objects are masked from 'package:dlookr':
##
##     kurtosis, skewness
```

```
sv <- svm(as.factor(target) ~., data = Model_data_Train, cost = 10, kernel = "sigmoid", scale =
T)
summary(sv)
```

```
##
## Call:
## svm(formula = as.factor(target) ~ ., data = Model_data_Train, cost = 10,
##     kernel = "sigmoid", scale = T)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  sigmoid
##        cost:  10
##      coef.0:  0
##
## Number of Support Vectors:  536
##
##  ( 176 268 92 )
##
##
## Number of Classes:  3
##
## Levels:
##  high_risk low_risk moderate_risk
```

Tuning the svm model

```
tune <- tune(svm, as.factor(target)~., data = Model_data_Train, kernel="sigmoid", ranges = list
(cost = c(0.1, 1,10,20,30,40,50)))
```

```
summary(tune)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##   0.1
##
## - best performance: 0.1184844
##
## - Detailed performance results:
##   cost     error dispersion
## 1  0.1 0.1184844 0.02665527
## 2  1.0 0.1515877 0.02358477
## 3 10.0 0.1704104 0.02555055
## 4 20.0 0.1700857 0.02451827
## 5 30.0 0.1700857 0.02461363
## 6 40.0 0.1697611 0.02436668
## 7 50.0 0.1700857 0.02451827
```

We then retrained the model with cost value of 0.1

```
sv_r <- svm(as.factor(target) ~., data = Model_data_Train, cost = 1, kernel = "sigmoid", scale = T)
summary(sv_r)
```

```
##
## Call:
## svm(formula = as.factor(target) ~ ., data = Model_data_Train, cost = 1,
##     kernel = "sigmoid", scale = T)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  sigmoid
##        cost:  1
##      coef.0:  0
##
## Number of Support Vectors:  570
##
##  ( 190 285 95 )
##
##
## Number of Classes:  3
##
## Levels:
##  high_risk low_risk moderate_risk
```

Model evaluation: Retrained vs trained

```
sv_pred <- predict(sv, newdata = Model_data_Test, type ="class")
acc <- confusionMatrix(as.factor(Model_data_Test$target), sv_pred)
acc
```

```
## Confusion Matrix and Statistics
##
##                Reference
## Prediction      high_risk low_risk moderate_risk
##   high_risk             4       10            20
##   low_risk             34      786            56
##   moderate_risk         0       57            61
##
## Overall Statistics
##
##                Accuracy : 0.8278
##                  95% CI : (0.8033, 0.8504)
##     No Information Rate : 0.8298
##     P-Value [Acc > NIR] : 0.5856
##
##                   Kappa : 0.3771
##
##  Mcnemar's Test P-Value : 3.068e-07
##
## Statistics by Class:
##
##                      Class: high_risk Class: low_risk Class: moderate_risk
## Sensitivity                  0.105263          0.9215              0.44526
## Specificity                  0.969697          0.4857              0.93603
## Pos Pred Value               0.117647          0.8973              0.51695
## Neg Pred Value               0.965795          0.5592              0.91648
## Prevalence                   0.036965          0.8298              0.13327
## Detection Rate               0.003891          0.7646              0.05934
## Detection Prevalence         0.033074          0.8521              0.11479
## Balanced Accuracy            0.537480          0.7036              0.69064
```

```
sv_rp <- predict(sv_r, newdata = Model_data_Test, type ="class")
acc_r <- confusionMatrix(as.factor(Model_data_Test$target), sv_rp)
acc_r
```

```
## Confusion Matrix and Statistics
##
##                Reference
## Prediction      high_risk low_risk moderate_risk
##   high_risk             3       11            20
##   low_risk             30      799            47
##   moderate_risk         0       57            61
##
## Overall Statistics
##
##                Accuracy : 0.8395
##                  95% CI : (0.8156, 0.8614)
##     No Information Rate : 0.8434
##     P-Value [Acc > NIR] : 0.6534
##
##                   Kappa : 0.3965
##
##  Mcnemar's Test P-Value : 1.545e-06
##
## Statistics by Class:
##
##                      Class: high_risk Class: low_risk Class: moderate_risk
## Sensitivity                  0.090909          0.9216              0.47656
## Specificity                  0.968844          0.5217              0.93667
## Pos Pred Value               0.088235          0.9121              0.51695
## Neg Pred Value               0.969819          0.5526              0.92637
## Prevalence                   0.032101          0.8434              0.12451
## Detection Rate               0.002918          0.7772              0.05934
## Detection Prevalence         0.033074          0.8521              0.11479
## Balanced Accuracy            0.529877          0.7217              0.70661
```

Conclusion

In summary, the Decision Tree model exhibited superior performance in accurately predicting medical diagnoses with three categories (high risk, low risk, and moderate) compared to the Support Vector Machine model, achieving higher overall accuracy, precision, and agreement with actual diagnoses.