Cardiovascular Disease Prediction

Data reading:

The data required is readily available in the open UCI Machine learning repository https://archive.ics.uci.edu/ml/datasets/heart+disease. But it is split up based on location and is in a custom format with the extension `.data`. We can read it as table and merge it. The column names are available in the file `heart-disease.names`.

Read the data

```
# Read all processed data

cleaveland_data <- read.table("./data/processed.cleveland.data", fileEncoding = "UTF-8", sep = ",")
hungarian_data <- read.table("./data/processed.hungarian.data", fileEncoding = "UTF-8", sep = ",")
switzerland_data <- read.table("./data/processed.switzerland.data", fileEncoding = "UTF-8", sep = ",")
va_data <- read.table("./data/processed.va.data", fileEncoding = "UTF-8", sep = ",")
```

Print the dimensions of read data:

```
print("Dimensions of individual datasets :")
```

```
## [1] "Dimensions of individual datasets :"
```

```
print(dim(cleaveland_data))
```

```
## [1] 303  14
```

```
print(dim(hungarian_data))
```

```
## [1] 294  14
```

```
print(dim(switzerland_data))
```

```
## [1] 123  14
```

```
print(dim(va_data))
```

```
## [1] 200  14
```

Concatinate the data and assign column names:

```
# Concat all the datasets
tmp1 <- rbind(cleaveland_data, hungarian_data)
tmp2 <- rbind(switzerland_data, va_data)
heart_data <- rbind(tmp1, tmp2)

# Column names from heart-disease.names file
colnames(heart_data) <- c("age","sex","cp","trestbps","chol","fbs","restecg","thalach","exang","oldpeak
summary(heart_data)
```

```
##       age             sex               cp         trestbps
##  Min.   :28.00   Min.   :0.0000   Min.   :1.00   Length:920
##  1st Qu.:47.00   1st Qu.:1.0000   1st Qu.:3.00   Class :character
##  Median :54.00   Median :1.0000   Median :4.00   Mode  :character
##  Mean   :53.51   Mean   :0.7891   Mean   :3.25
##  3rd Qu.:60.00   3rd Qu.:1.0000   3rd Qu.:4.00
##  Max.   :77.00   Max.   :1.0000   Max.   :4.00
##      chol               fbs              restecg            thalach
##  Length:920         Length:920         Length:920         Length:920
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##     exang             oldpeak             slope                ca
##  Length:920         Length:920         Length:920         Length:920
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##      thal                goal
##  Length:920         Min.   :0.0000
##  Class :character   1st Qu.:0.0000
##  Mode  :character   Median :1.0000
##                     Mean   :0.9957
##                     3rd Qu.:2.0000
##                     Max.   :4.0000
```

```
print("Dimensions of combined data :")
```

```
## [1] "Dimensions of combined data :"
```

```
print(dim(heart_data))
```

```
## [1] 920  14
```

Remove all unnecessary columns with ? or :

```r
heart_data [heart_data == "?"] <- NA
heart_data <- drop_na(heart_data)

# Check if we still have any na values
apply(heart_data,2, function(x) any(is.na(x)))
```

```
##      age      sex       cp trestbps     chol      fbs  restecg  thalach
##    FALSE    FALSE    FALSE    FALSE    FALSE    FALSE    FALSE    FALSE
##    exang  oldpeak    slope       ca     thal     goal
##    FALSE    FALSE    FALSE    FALSE    FALSE    FALSE
```

```r
# After removal of all the data we are down to 299 rows
dim(heart_data)
```

```
## [1] 299  14
```

See if the data types are okay

```r
# print the data types
print(sapply(heart_data, class))
```

```
##         age         sex          cp    trestbps        chol         fbs
##   "numeric"   "numeric"   "numeric" "character" "character" "character"
##     restecg     thalach       exang     oldpeak       slope          ca
## "character" "character" "character" "character" "character" "character"
##        thal        goal
## "character"   "integer"
```

Fix the data types

```r
# Data types are wrong, should update it based on data available from heart-disease.names
# Age should be a number
heart_data$age <- as.numeric(heart_data$age)

# Sex should be a factor (1 = male; 0 = female)
heart_data$sex <- as.factor(heart_data$sex)

# cp - chest pain should be a factor
# Value 1: typical angina
# Value 2: atypical angina
# Value 3: non-anginal pain
# Value 4: asymptomatic
heart_data$cp <- as.factor(heart_data$cp)

# trestbps - resting blood pressure
heart_data$trestbps <- as.numeric(heart_data$trestbps)

# chol - serum cholestoral in mg/dl
```

```r
heart_data$chol <- as.numeric(heart_data$chol)

# fbs - If fasting blood sugar > 120 mg/dl,   (1 = true; 0 = false)
heart_data$fbs <- as.factor(heart_data$fbs)

# restecg - resting electrocardiographic results
# Value 0: normal
# Value 1: having ST-T wave abnormality (T wave inversions and/or ST elevation or depression of > 0.05
# Value 2: showing probable or definite left ventricular hypertrophy by Estes' criteria
heart_data$restecg <- as.factor(heart_data$restecg)

# thalach: maximum heart rate achieved
heart_data$thalach <- as.numeric(heart_data$thalach)

# exang: exercise induced angina (1 = yes; 0 = no)
heart_data$exang <- as.factor(heart_data$exang)

# oldpeak = ST depression induced by exercise relative to rest
heart_data$oldpeak <- as.numeric(heart_data$oldpeak)

# slope: the slope of the peak exercise ST segment
# Value 1: upsloping
# Value 2: flat
# Value 3: downsloping
heart_data$slope <- as.factor(heart_data$slope)

# ca: number of major vessels (0-3) colored by flourosopy
heart_data$ca <- as.numeric(heart_data$ca)

# thal: 3 = normal; 6 = fixed defect; 7 = reversable defect
heart_data$thal <- as.factor(as.integer(heart_data$thal))

# goal: It distinguish presence (values 1,2,3,4) from absence (value 0)
heart_data$goal <- as.factor(heart_data$goal)

print("After manual updates of datatype")
```

```
## [1] "After manual updates of datatype"
```

```r
# print the data types
print(sapply(heart_data, class))
```

```
##       age       sex        cp  trestbps      chol       fbs   restecg   thalach
## "numeric"  "factor"  "factor" "numeric" "numeric"  "factor"  "factor" "numeric"
##     exang   oldpeak     slope        ca      thal      goal
##  "factor" "numeric"  "factor" "numeric"  "factor"  "factor"
```

```r
summary(heart_data)
```

```
##       age         sex     cp         trestbps          chol         fbs
##  Min.   :29.00   0: 96   1: 23   Min.   : 94.0   Min.   :100.0   0:256
##  1st Qu.:48.00   1:203   2: 49   1st Qu.:120.0   1st Qu.:211.0   1: 43
##  Median :56.00          3: 83   Median :130.0   Median :242.0  
##  Mean   :54.52          4:144   Mean   :131.7   Mean   :246.8  
##  3rd Qu.:61.00                  3rd Qu.:140.0   3rd Qu.:275.5  
##  Max.   :77.00                  Max.   :200.0   Max.   :564.0  
##  restecg    thalach      exang      oldpeak      slope         ca        
##  0:149   Min.   : 71.0   0:200   Min.   :0.000   1:139   Min.   :0.0000  
##  1:  4   1st Qu.:132.5   1: 99   1st Qu.:0.000   2:139   1st Qu.:0.0000  
##  2:146   Median :152.0           Median :0.800   3: 21   Median :0.0000  
##          Mean   :149.3           Mean   :1.059           Mean   :0.6722  
##          3rd Qu.:165.5           3rd Qu.:1.600           3rd Qu.:1.0000  
##          Max.   :202.0           Max.   :6.200           Max.   :3.0000  
##  thal    goal   
##  3:164   0:160  
##  6: 18   1: 56  
##  7:117   2: 35  
##          3: 35  
##          4: 13  
## 
```

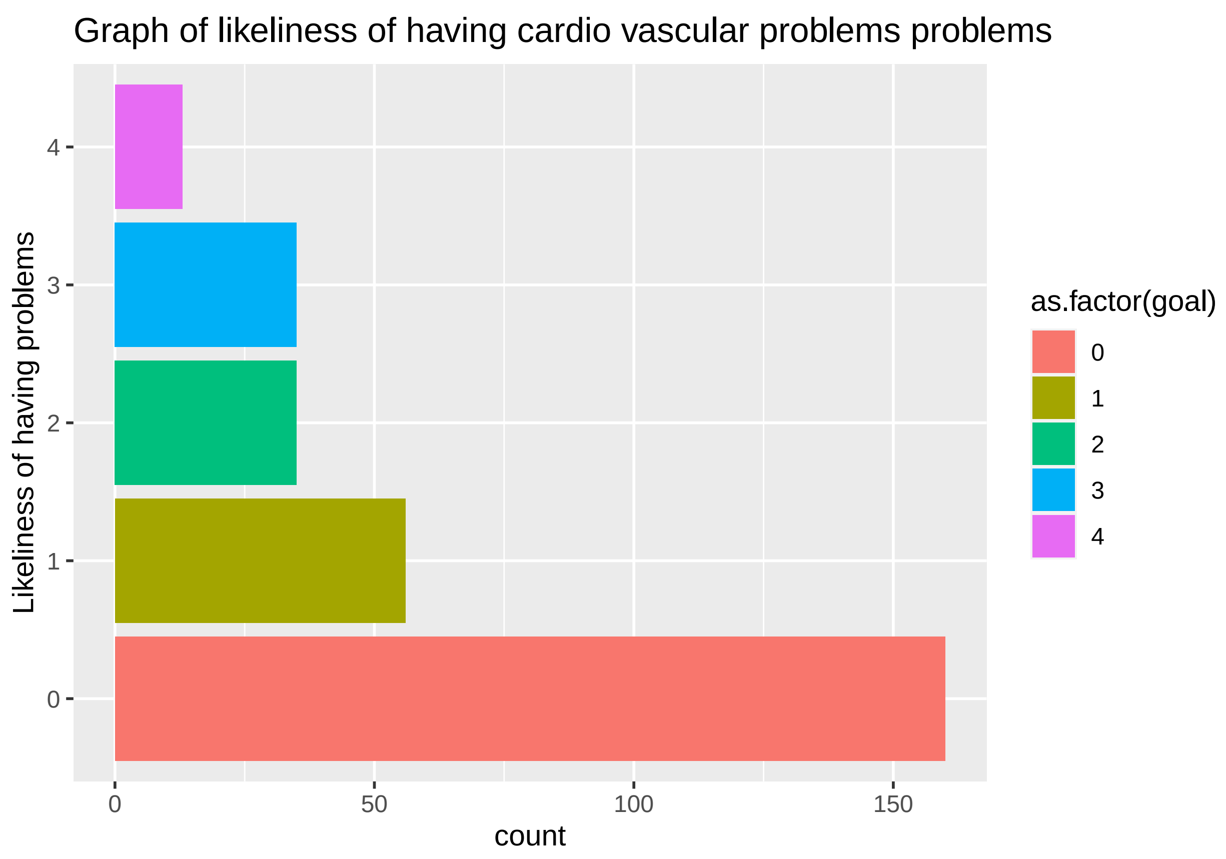
Write the data as csv to local machine:

```
write.csv(heart_data,"./data/heart_data.csv", row.names = FALSE)
```

Data exploration:

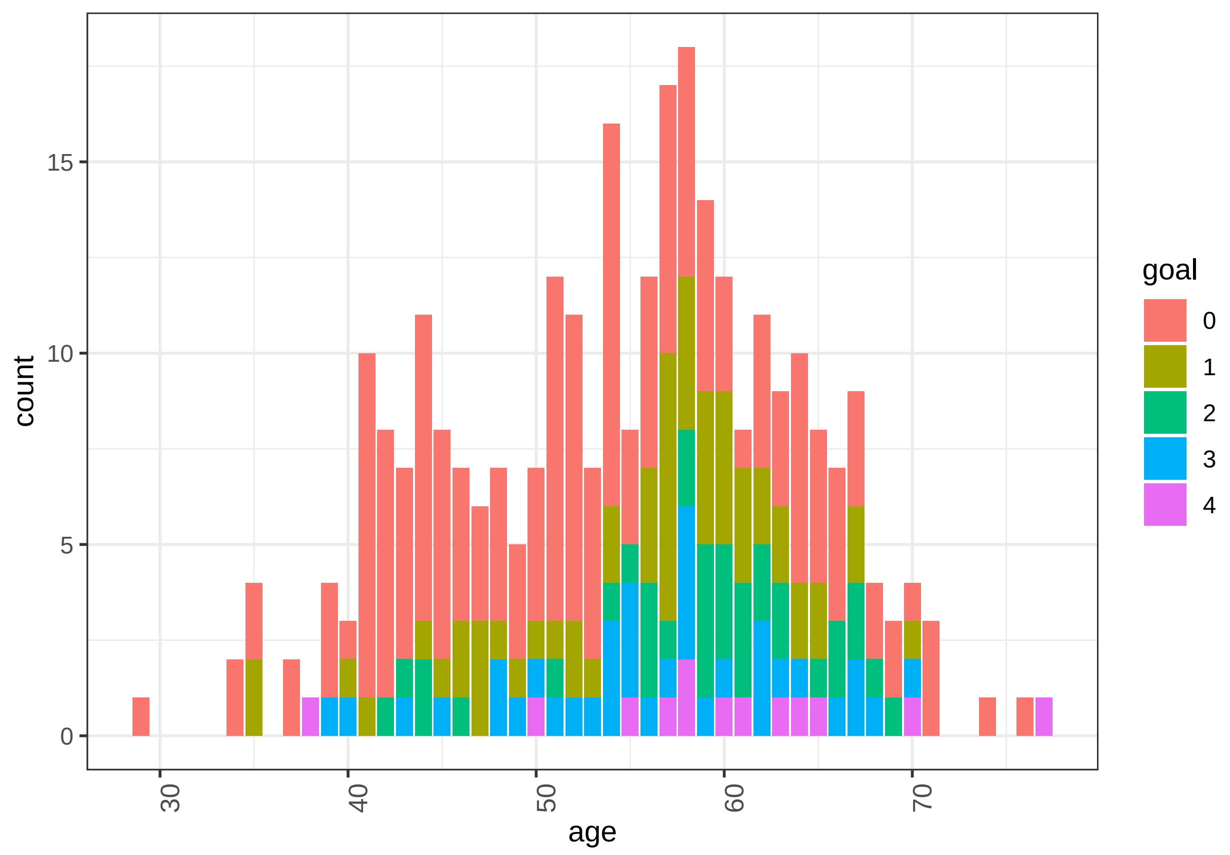
Plot a graph on likeliness of people having a cardio-vascular problems with 0 as abscense and (1,2,3,4) having problems.

```
ggplot(heart_data, aes(x=as.factor(goal), fill=as.factor(goal) )) +
  geom_bar() +
  xlab("Likeliness of having problems") +
  ggtitle("Graph of likeliness of having cardio vascular problems problems") +
  coord_flip()
```



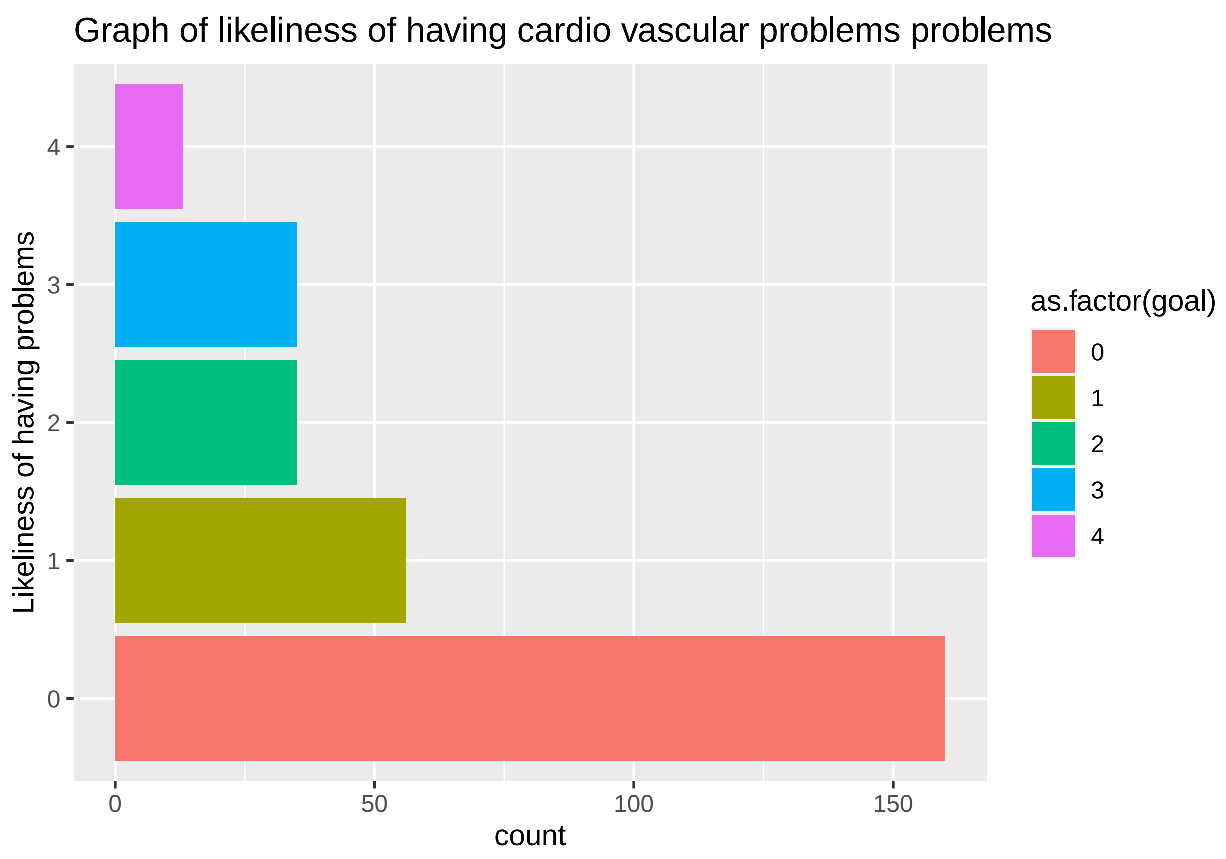
```
heart_data %>% group_by(age, goal) %>% summarise(count = n()) %>%
  ggplot() + geom_bar(aes(age, count,   fill = as.factor(goal)), stat = "Identity") +
  theme_bw() +
  theme(axis.text.x = element_text(angle = 90, size = 10)) +
  ylab("count") + xlab("age") + labs(fill = "goal")
```

```
## `summarise()` has grouped output by 'age'. You can override using the `.groups`
## argument.
```



Plot a graph on likeliness of people having a cardio-vascular problems with 0 as abscense and (1,2,3,4) having problems.

```
ggplot(heart_data, aes(x=as.factor(goal), fill=as.factor(goal) )) +
  geom_bar() +
  xlab("Likeliness of having problems") +
  ggtitle("Graph of likeliness of having cardio vascular problems problems") +
  coord_flip()
```

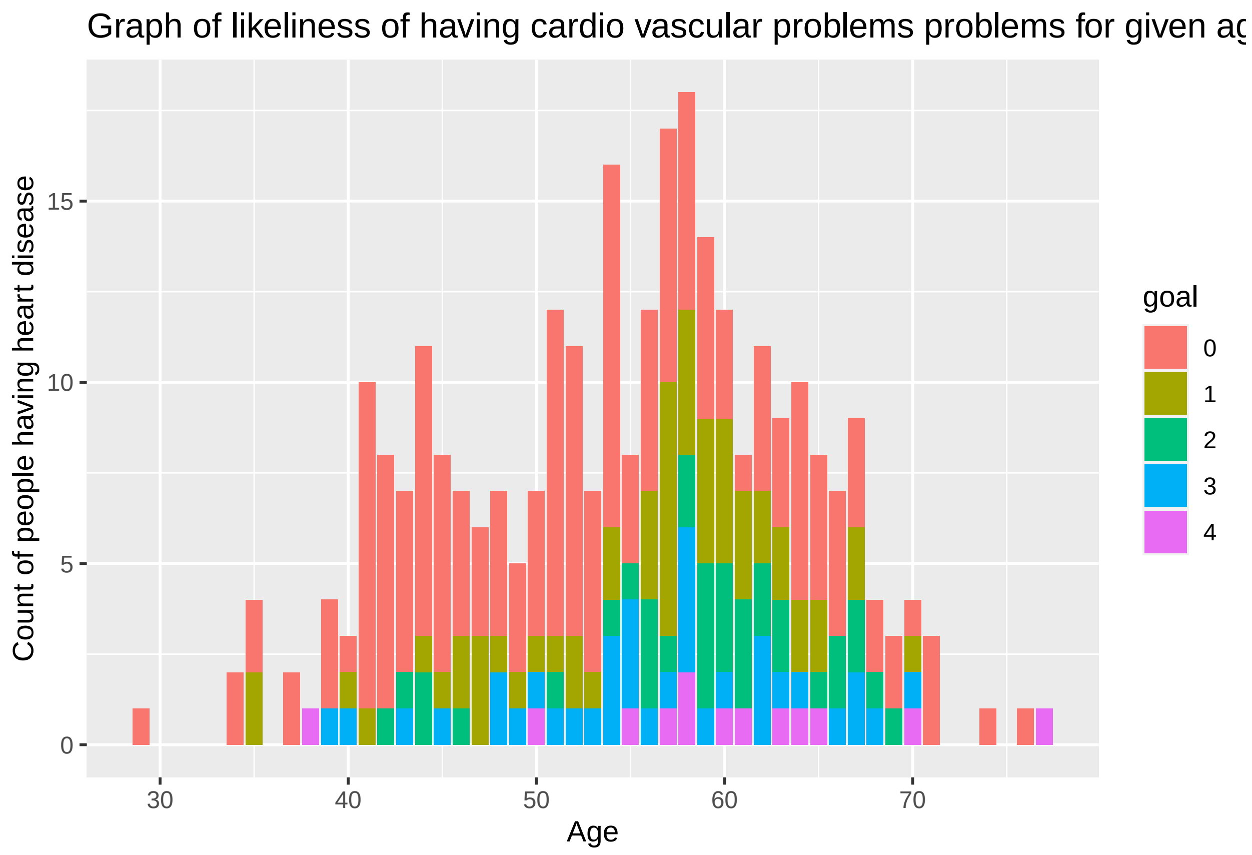


Graph of likeliness of having cardio vascular problems problems for given age

```
grouped_data <- group_by(heart_data, age, goal)
heart_data_summary <- summarise(grouped_data, count = n())
```

```
## ‘summarise()‘ has grouped output by ’age’. You can override using the ‘.groups‘
## argument.
```

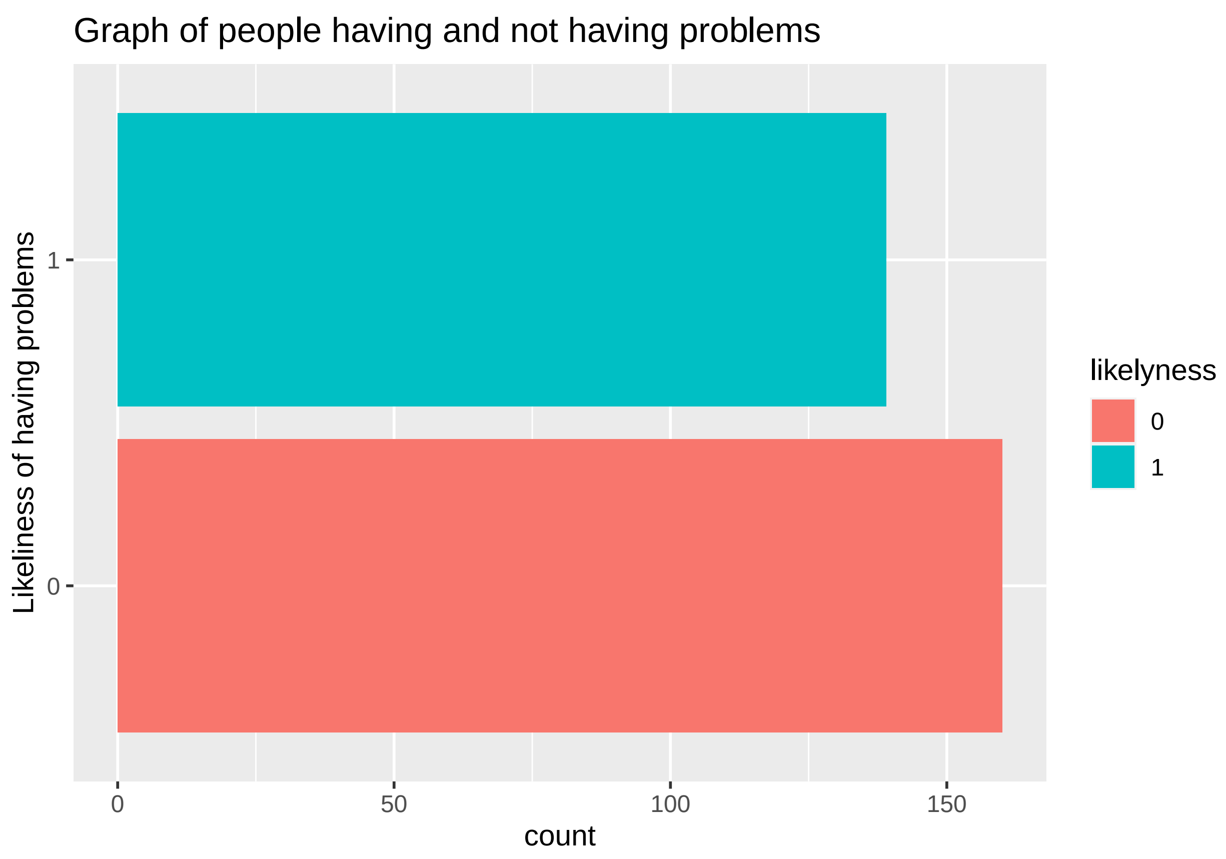
```
ggplot(heart_data_summary) +
geom_bar(aes(age, count, fill = goal), stat = "Identity") +
ylab("Count of people having heart disease") +
ggtitle("Graph of likeliness of having cardio vascular problems problems for given age") +
xlab("Age")
```



Plot a graph on likeliness of people having a cardio-vascular problems with 0 as abscense and 1 as having problems.

```r
likelyness <- as.factor(ifelse(heart_data$goal == 0,0,1))

ggplot(heart_data, aes(x=likelyness, fill=likelyness)) +
  geom_bar() +
  xlab("Likeliness of having problems") +
  ggtitle("Graph of people having and not having problems") +
  coord_flip()
```



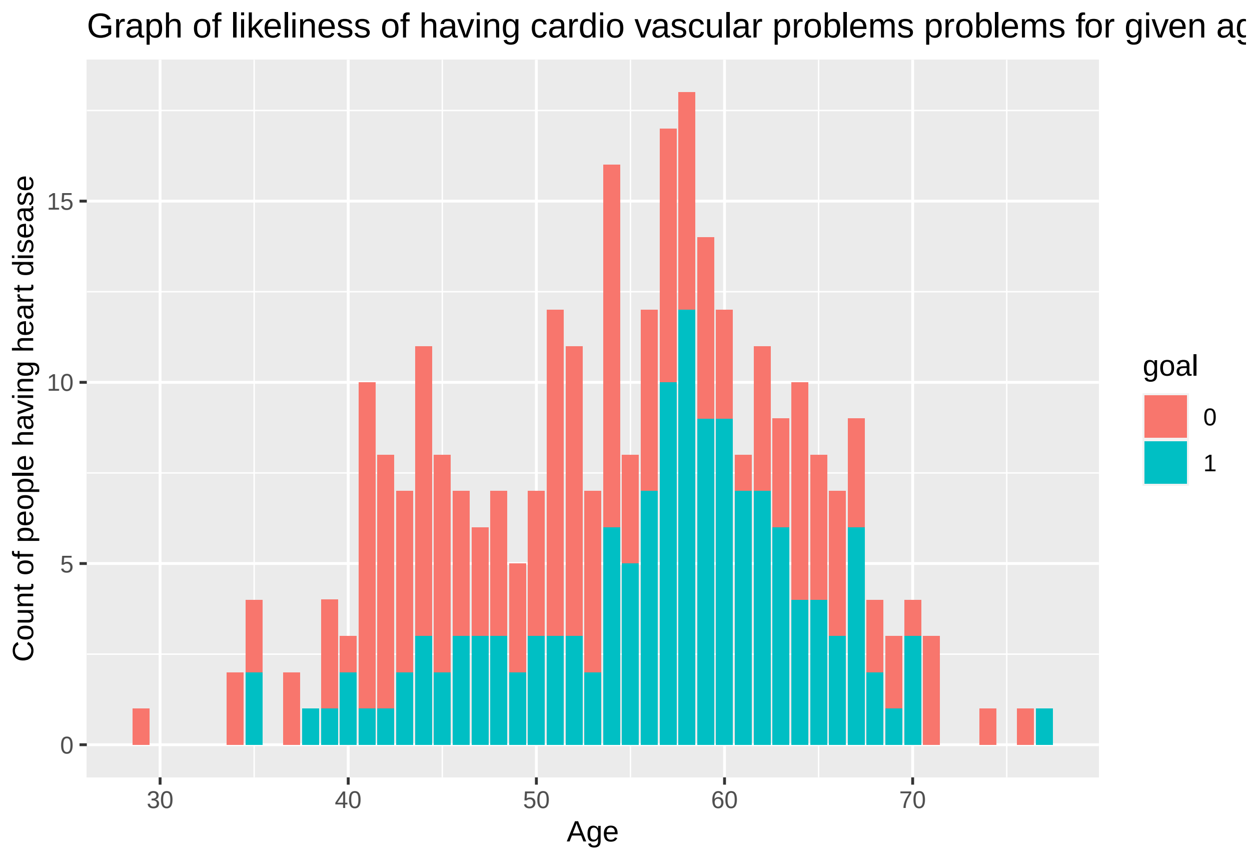
Graph of people having and not having problems for given age

```
tmp_heart_data <- heart_data
tmp_heart_data$goal <- as.factor(ifelse(heart_data$goal == 0,0,1))


grouped_data <- group_by(tmp_heart_data, age, goal)
heart_data_summary <- summarise(grouped_data, count = n())
```

```
## `summarise()` has grouped output by 'age'. You can override using the `.groups`
## argument.
```

```
ggplot(heart_data_summary) +
geom_bar(aes(age, count, fill = goal), stat = "Identity") +
ylab("Count of people having heart disease") +
ggtitle("Graph of likeliness of having cardio vascular problems problems for given age") +
xlab("Age")
```

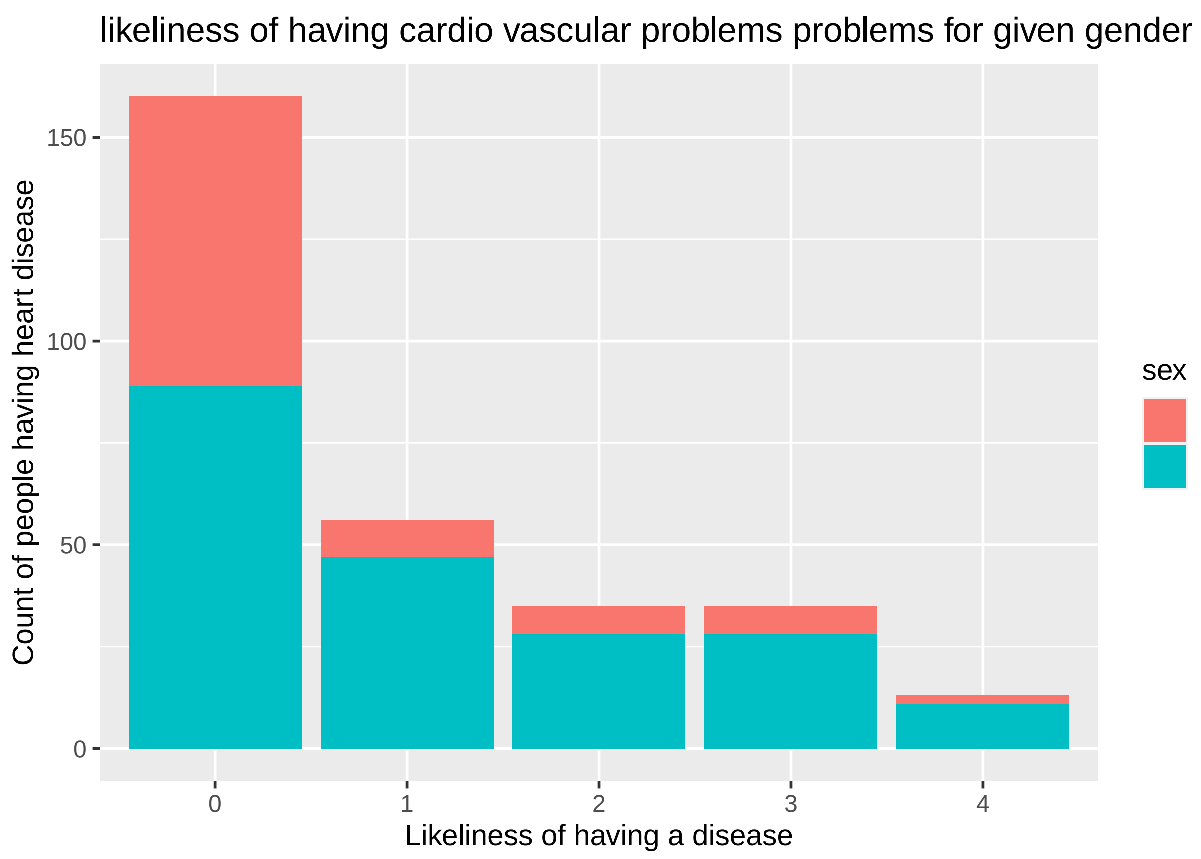


likeliness of having cardio vascular problems problems for given gender

```
grouped_data <- group_by(heart_data, sex, goal)
heart_data_summary <- summarise(grouped_data, count = n())
```

```
## ‘summarise()‘ has grouped output by ’sex’. You can override using the ‘.groups‘
## argument.
```

```
ggplot(heart_data_summary) +
geom_bar(aes(goal, count, fill = sex), stat = "Identity") +
ylab("Count of people having heart disease") +
ggtitle("likeliness of having cardio vascular problems problems for given gender") +
xlab("Likeliness of having a disease")
```

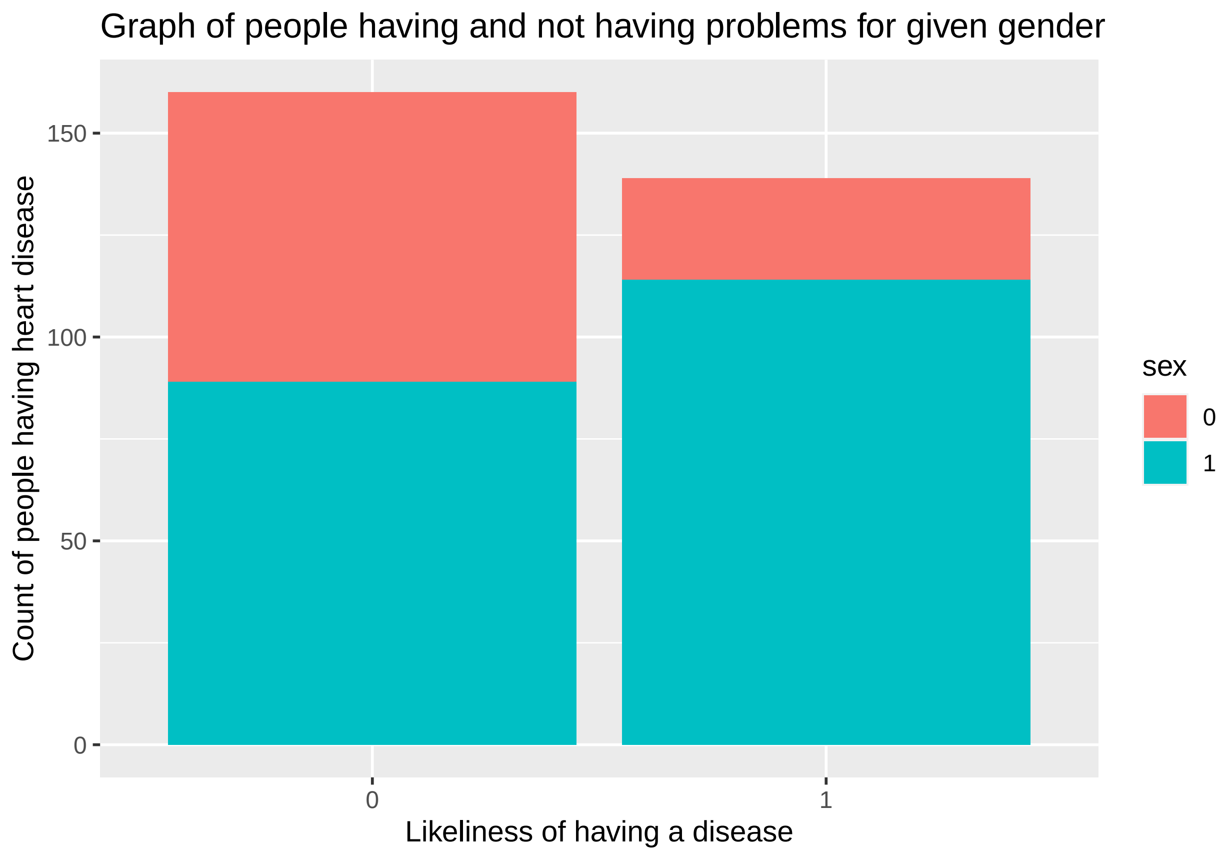


Graph of people having and not having problems for given gender

```
grouped_data <- group_by(tmp_heart_data, sex, goal)
heart_data_summary <- summarise(grouped_data, count = n())
```

```
## ‘summarise()‘ has grouped output by ’sex’. You can override using the ‘.groups‘
## argument.
```

```
ggplot(heart_data_summary) +
geom_bar(aes(goal, count, fill = sex), stat = "Identity") +
ylab("Count of people having heart disease") +
ggtitle("Graph of people having and not having problems for given gender") +
xlab("Likeliness of having a disease")
```



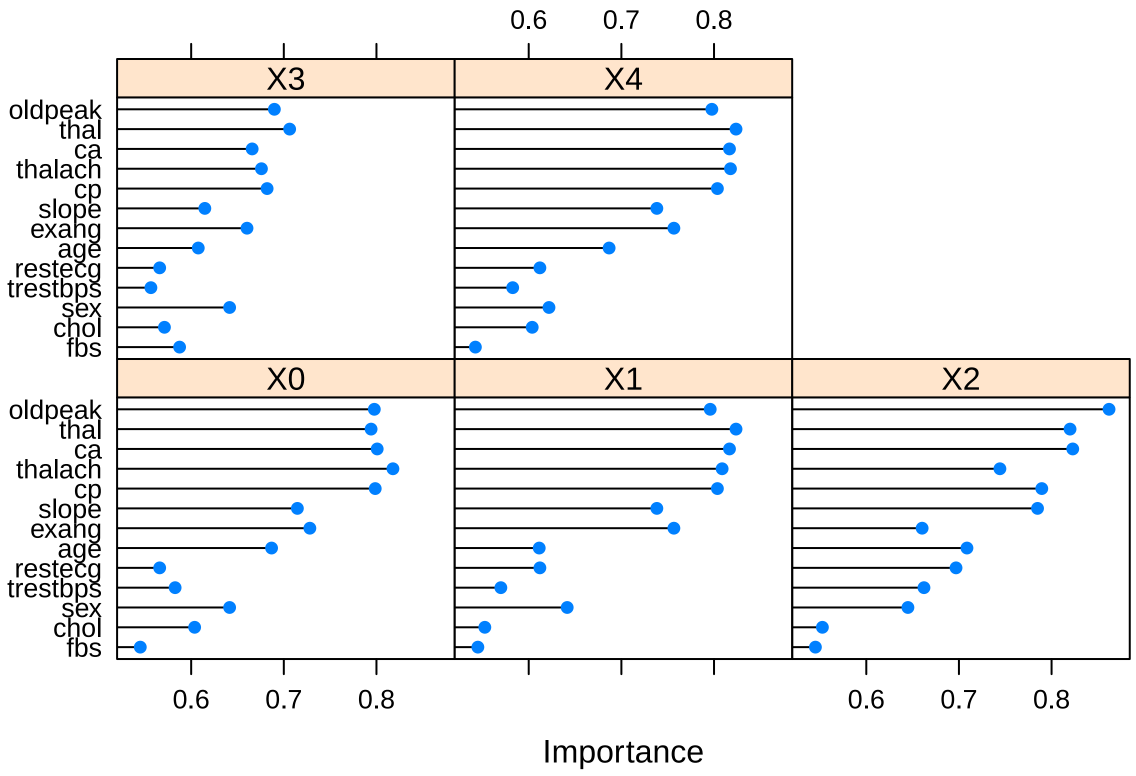
Feature selection:

Rank the variables based on their importance using Learning Vector Quantization We can use the caret library where we can build LVQ model and use varImp to see the variable importance. Here we see that the `thal` is the most important where as the `fbs`, `chol`, `restecg` has the least values.

```
lvq.model <- train(goal ~ .,
                  data=heart_data,
                  method="lvq",
                  preProcess="scale",
                  trControl=trainControl(method="repeatedcv", number=10, repeats=3))

lvq.res <- varImp(lvq.model, scale=FALSE)

plot(lvq.res)
```



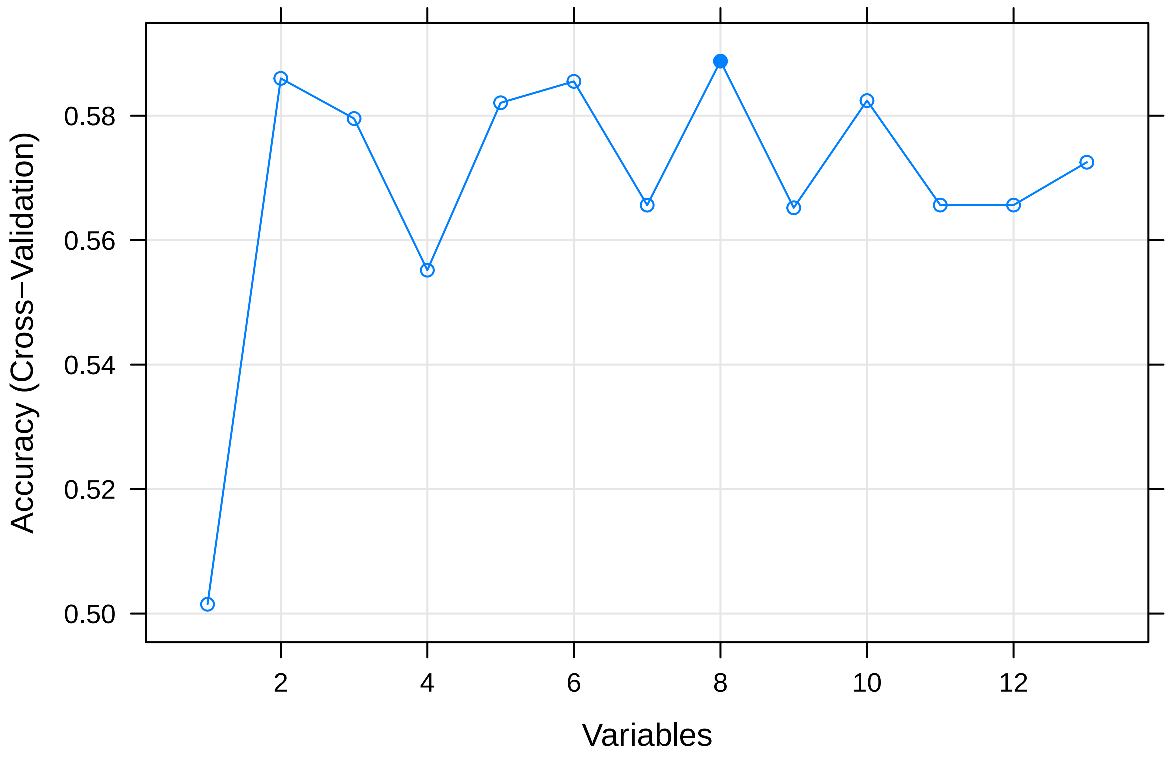
Use Recursive Feature Elimination to try and eliminate some noise

We can use the RFE to automatically eliminate features that are the least important. As we can see fbs, chol, restecg values whose varImp is the least have been removed.

```
rfe.res <- rfe(heart_data[,1:13],
               heart_data[,14],
               sizes=c(1:13),
               rfeControl=rfeControl(functions=rfFuncs, method="cv", number=10))

rfe.predictors <- predictors(rfe.res)

plot(rfe.res, type=c("g", "o"))
```



```r
# update the heart data based on the predictors
heart_data <- subset(heart_data, select = append(rfe.predictors, 'goal' ,after=1))

str(heart_data)
```

```
## 'data.frame':    299 obs. of  9 variables:
##  $ ca     : num  0 3 2 0 0 0 2 0 1 0 ...
##  $ goal   : Factor w/ 5 levels "0","1","2","3",..: 1 3 2 1 1 1 4 1 3 2 ...
##  $ thal   : Factor w/ 3 levels "3","6","7": 2 1 3 1 1 1 1 1 3 3 ...
##  $ cp     : Factor w/ 4 levels "1","2","3","4": 1 4 4 3 2 2 4 4 4 4 ...
##  $ oldpeak: num  2.3 1.5 2.6 3.5 1.4 0.8 3.6 0.6 1.4 3.1 ...
##  $ thalach: num  150 108 129 187 172 178 160 163 147 155 ...
##  $ exang  : Factor w/ 2 levels "0","1": 1 2 2 1 1 1 1 2 1 2 ...
##  $ slope  : Factor w/ 3 levels "1","2","3": 3 2 2 3 1 1 3 1 2 3 ...
##  $ sex    : Factor w/ 2 levels "0","1": 2 2 2 2 1 2 1 1 2 2 ...
```

Prediction:

We will be using various classification algorithms to help predict cardio vascular diseases. Compare accuracy and come to conclusion:

- K Nearest Neighbours
- Support Vector Machines
- Random Forest

- Gradient Boosting Machines
- Linear Discriminant Analysis
- Quadrant Discriminant Analysis

We can use the caret library which provides easy access to all the above algorithms

Let us use a simplify the goal as either 0 (absense) or 1 (presence)

```
heart_data$goal <- as.factor(ifelse(heart_data$goal == 0,0,1))

str(heart_data)
```

```
## 'data.frame':    299 obs. of  9 variables:
##  $ ca     : num  0 3 2 0 0 0 2 0 1 0 ...
##  $ goal   : Factor w/ 2 levels "0","1": 1 2 2 1 1 1 2 1 2 2 ...
##  $ thal   : Factor w/ 3 levels "3","6","7": 2 1 3 1 1 1 1 1 3 3 ...
##  $ cp     : Factor w/ 4 levels "1","2","3","4": 1 4 4 3 2 2 4 4 4 4 ...
##  $ oldpeak: num  2.3 1.5 2.6 3.5 1.4 0.8 3.6 0.6 1.4 3.1 ...
##  $ thalach: num  150 108 129 187 172 178 160 163 147 155 ...
##  $ exang  : Factor w/ 2 levels "0","1": 1 2 2 1 1 1 1 2 1 2 ...
##  $ slope  : Factor w/ 3 levels "1","2","3": 3 2 2 3 1 1 3 1 2 3 ...
##  $ sex    : Factor w/ 2 levels "0","1": 2 2 2 2 1 2 1 1 2 2 ...
```

First let us divide the data into two different sets:

```
set.seed(2022)

# Divide the dataset 5:5
split <- sample.split(heart_data, SplitRatio = 0.7)

training_data <- subset(heart_data, split == "TRUE")
dim(training_data)
```

```
## [1] 198   9
```

```
validation_data <- subset(heart_data, split == "FALSE")
dim(validation_data)
```

```
## [1] 101   9
```

K Nearest Neighbours

```
knn.model <- train(goal ~ .,
                data = training_data,
                method = "knn",
                preProcess = c("center","scale"),
                trControl = trainControl(method = "cv", verboseIter = FALSE, number = 2),
                tuneGrid = expand.grid(k = 1:20))

knn.res <- predict(knn.model, newdata = validation_data )
knn.confusion_matrix <- confusionMatrix(knn.res, validation_data$goal )

knn.confusion_matrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 41  5
##          1  8 47
##
##                Accuracy : 0.8713
##                  95% CI : (0.79, 0.9296)
##     No Information Rate : 0.5149
##     P-Value [Acc > NIR] : 3.373e-14
##
##                   Kappa : 0.7419
##
##  Mcnemar's Test P-Value : 0.5791
##
##             Sensitivity : 0.8367
##             Specificity : 0.9038
##          Pos Pred Value : 0.8913
##          Neg Pred Value : 0.8545
##              Prevalence : 0.4851
##          Detection Rate : 0.4059
##    Detection Prevalence : 0.4554
##       Balanced Accuracy : 0.8703
##
##        'Positive' Class : 0
##
```

Support Vector Machines

```
svm.model <- train(goal ~ .,
                   data = training_data,
                   method = "svmLinear",
                   preProcess = c("center","scale"),
                   tuneGrid = expand.grid(C = c(0.01, 0.1, 1, 10, 20)),
                   trControl = trainControl(method = "cv", verboseIter = FALSE, number = 2))

svm.res <- predict(svm.model,
                     newdata = validation_data)

svm.confusion_matrix <- confusionMatrix(svm.res, validation_data$goal)

svm.confusion_matrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 44  5
##          1  5 47
##
##                Accuracy : 0.901
##                   95% CI : (0.8254, 0.9515)
##     No Information Rate : 0.5149
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.8018
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.8980
##             Specificity : 0.9038
##          Pos Pred Value : 0.8980
##          Neg Pred Value : 0.9038
##              Prevalence : 0.4851
##          Detection Rate : 0.4356
##    Detection Prevalence : 0.4851
##       Balanced Accuracy : 0.9009
##
##        'Positive' Class : 0
##
```

Random Forest

```
rf.model <- train(goal ~ .,
                method = "rf",
                data = training_data,
                ntree = 20,
                trControl = trainControl(method = "cv", number = 2, verboseIter = FALSE),
                tuneGrid = data.frame(mtry = c(1:10)))

rf.res <- predict(rf.model, newdata = validation_data)

rf.confusion_matrix <- confusionMatrix(rf.res, validation_data$goal)

rf.confusion_matrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 46 10
##          1  3 42
##
##                Accuracy : 0.8713
##                  95% CI : (0.79, 0.9296)
##     No Information Rate : 0.5149
##     P-Value [Acc > NIR] : 3.373e-14
##
##                   Kappa : 0.7434
##
##  Mcnemar's Test P-Value : 0.09609
##
##             Sensitivity : 0.9388
##            Specificity : 0.8077
##         Pos Pred Value : 0.8214
##         Neg Pred Value : 0.9333
##             Prevalence : 0.4851
##         Detection Rate : 0.4554
##   Detection Prevalence : 0.5545
##      Balanced Accuracy : 0.8732
##
##       'Positive' Class : 0
##
```

Gradient Boosting Machines

```
gbm.model <- train(goal ~ .,
                 method = "gbm",
                 verbose = FALSE,
                 data = training_data,
                 trControl = trainControl(method = "cv", number = 2, verboseIter = FALSE),
                 tuneGrid = expand.grid(interaction.depth = seq(5, 30, 5),
                        n.trees = seq(5, 50, 5),
                        shrinkage = c(0.1:0.5),
                        n.minobsinnode = 10))

gbm.res <- predict(gbm.model, newdata = validation_data)

gbm.confusion_matrix <- confusionMatrix(gbm.res, validation_data$goal)

gbm.confusion_matrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 42  5
##          1  7 47
##
##                Accuracy : 0.8812
##                  95% CI : (0.8017, 0.9371)
##     No Information Rate : 0.5149
##     P-Value [Acc > NIR] : 5.149e-15
##
##                   Kappa : 0.7619
##
##  Mcnemar's Test P-Value : 0.7728
##
##             Sensitivity : 0.8571
##             Specificity : 0.9038
##          Pos Pred Value : 0.8936
##          Neg Pred Value : 0.8704
##              Prevalence : 0.4851
##          Detection Rate : 0.4158
##    Detection Prevalence : 0.4653
##       Balanced Accuracy : 0.8805
##
##        'Positive' Class : 0
##
```

Linear Discriminant Analysis

```
lda.model <- train(goal ~ .,
                   method = "lda",
                   data = training_data)

lda.res <- predict(lda.model, validation_data)
lda.confusion_matrix <- confusionMatrix(lda.res, validation_data$goal)

lda.confusion_matrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 43  5
##          1  6 47
##
##                Accuracy : 0.8911
##                  95% CI : (0.8135, 0.9444)
##     No Information Rate : 0.5149
##     P-Value [Acc > NIR] : 7.177e-16
##
##                   Kappa : 0.7819
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.8776
##             Specificity : 0.9038
##          Pos Pred Value : 0.8958
##          Neg Pred Value : 0.8868
##              Prevalence : 0.4851
##          Detection Rate : 0.4257
##    Detection Prevalence : 0.4752
##       Balanced Accuracy : 0.8907
##
##        'Positive' Class : 0
##
```

Quadrant Discriminant Analysis

```
qda.model <- train(goal ~ .,
                   method = "qda",
                   data = training_data)
```

```
## Warning: model fit failed for Resample14: parameter=none Error in qda.default(x, grouping, ...) : ra

## Warning: model fit failed for Resample18: parameter=none Error in qda.default(x, grouping, ...) : ra

## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo, :
## There were missing values in resampled performance measures.
```

```
qda.res <- predict(qda.model, validation_data)
qda.confusion_matrix <- confusionMatrix(qda.res, validation_data$goal)

qda.confusion_matrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 44  8
##          1  5 44
##
##                Accuracy : 0.8713
##                  95% CI : (0.79, 0.9296)
##     No Information Rate : 0.5149
##     P-Value [Acc > NIR] : 3.373e-14
##
##                   Kappa : 0.7428
##
##  Mcnemar's Test P-Value : 0.5791
##
##             Sensitivity : 0.8980
##             Specificity : 0.8462
##          Pos Pred Value : 0.8462
##          Neg Pred Value : 0.8980
##              Prevalence : 0.4851
##          Detection Rate : 0.4356
##    Detection Prevalence : 0.5149
##       Balanced Accuracy : 0.8721
##
##        'Positive' Class : 0
##
```

Conclusion

Initially the results were always less than 80%, but as we kept on cleaning up of data, did feature elimination, performed cross validation and did some hyper-parameter tuning with tuneGrid the results got better with some models even getting more than 90% accuracy.