

R Project : Heat attack

Load libraries

```
library(tidyverse)
library(caret)
library(RColorBrewer)
library(corrplot)
```

Data preparation

Import dataset

```
df <- read.csv("heart.csv")
```

Check df's structure

```
str(df)

## 'data.frame':    303 obs. of  14 variables:
## $ age      : int  63 37 41 56 57 57 56 44 52 57 ...
## $ sex      : int  1 1 0 1 0 1 0 1 1 1 ...
## $ cp       : int  3 2 1 1 0 0 1 1 2 2 ...
## $ trtbps   : int  145 130 130 120 120 140 140 120 172 150 ...
## $ chol     : int  233 250 204 236 354 192 294 263 199 168 ...
## $ fbs      : int  1 0 0 0 0 0 0 0 1 0 ...
## $ restecg  : int  0 1 0 1 1 1 0 1 1 1 ...
## $ thalachh : int  150 187 172 178 163 148 153 173 162 174 ...
## $ exng     : int  0 0 0 0 1 0 0 0 0 0 ...
## $ oldpeak  : num  2.3 3.5 1.4 0.8 0.6 0.4 1.3 0 0.5 1.6 ...
## $ slp      : int  0 0 2 2 2 1 1 2 2 2 ...
## $ caa      : int  0 0 0 0 0 0 0 0 0 0 ...
## $ thall    : int  1 2 2 2 2 1 2 3 3 2 ...
## $ output   : int  1 1 1 1 1 1 1 1 1 1 ...
```

Check missing value

```
colSums(is.na(df))

##      age      sex      cp  trtbps      chol      fbs  restecg  thalachh
##      0        0        0        0        0        0        0        0
##  exng  oldpeak    slp      caa    thall  output
##      0        0        0        0        0        0
```

Check and remove duplicated value

```
df[duplicated(df)|duplicated(df,fromLast = TRUE),]  
  
##      age sex cp trtbps chol fbs restecg thalachh exng oldpeak slp caa thall  
## 164   38  1  2   138  175  0         1       173   0       0   2   4     2  
## 165   38  1  2   138  175  0         1       173   0       0   2   4     2  
##      output  
## 164         1  
## 165         1  
  
df <- df[!duplicated(df),]
```

Data transformation

```
df$sex <- factor(df$sex,  
                 levels = c(0,1),  
                 labels = c("F","M"))  
  
df$output <- factor(df$output,  
                    levels = c(0,1),  
                    labels = c("0","1"))  
df$cp <- as.factor(df$cp)  
df$restecg <- as.factor(df$restecg)
```

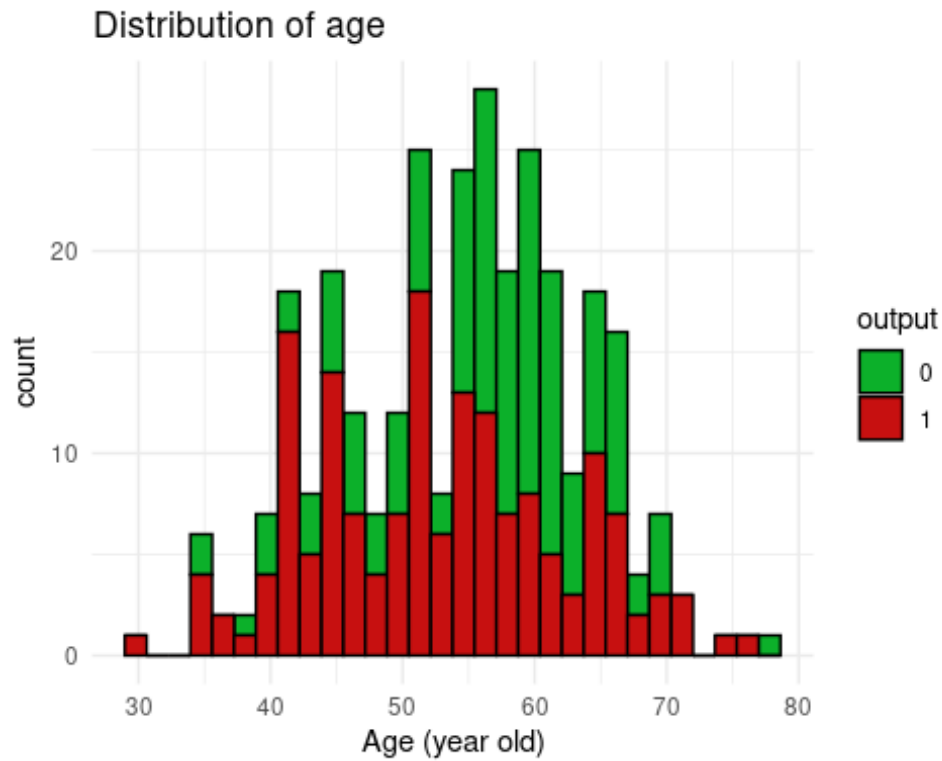
Summary of dataset

```
summary(df)  
  
##      age      sex      cp      trtbps      chol  
##  Min.   :29.00  F: 96    0:143  Min.    : 94.0  Min.    :126.0  
## 1st Qu.:48.00  M:206  1: 50  1st Qu.:120.0  1st Qu.:211.0  
## Median :55.50          2: 86  Median :130.0  Median :240.5  
## Mean   :54.42          3: 23  Mean   :131.6  Mean   :246.5  
## 3rd Qu.:61.00          3rd Qu.:140.0  3rd Qu.:274.8  
## Max.   :77.00          Max.   :200.0  Max.   :564.0  
##      fbs      restecg      thalachh      exng      oldpeak  
##  Min.    :0.000  0:147  Min.    : 71.0  Min.    :0.0000  Min.    :0.000  
## 1st Qu.:0.000  1:151  1st Qu.:133.2  1st Qu.:0.0000  1st Qu.:0.000  
## Median :0.000  2:  4  Median :152.5  Median :0.0000  Median :0.800  
## Mean   :0.149          Mean   :149.6  Mean   :0.3278  Mean   :1.043  
## 3rd Qu.:0.000          3rd Qu.:166.0  3rd Qu.:1.0000  3rd Qu.:1.600  
## Max.   :1.000          Max.   :202.0  Max.   :1.0000  Max.   :6.200  
##      slp      caa      thall      output  
##  Min.    :0.000  Min.    :0.0000  Min.    :0.000  0:138  
## 1st Qu.:1.000  1st Qu.:0.0000  1st Qu.:2.000  1:164  
## Median :1.000  Median :0.0000  Median :2.000  
## Mean   :1.397  Mean   :0.7185  Mean   :2.315  
## 3rd Qu.:2.000  3rd Qu.:1.0000  3rd Qu.:3.000  
## Max.   :2.000  Max.   :4.0000  Max.   :3.000
```

Exploratory data analysis

Age distribution

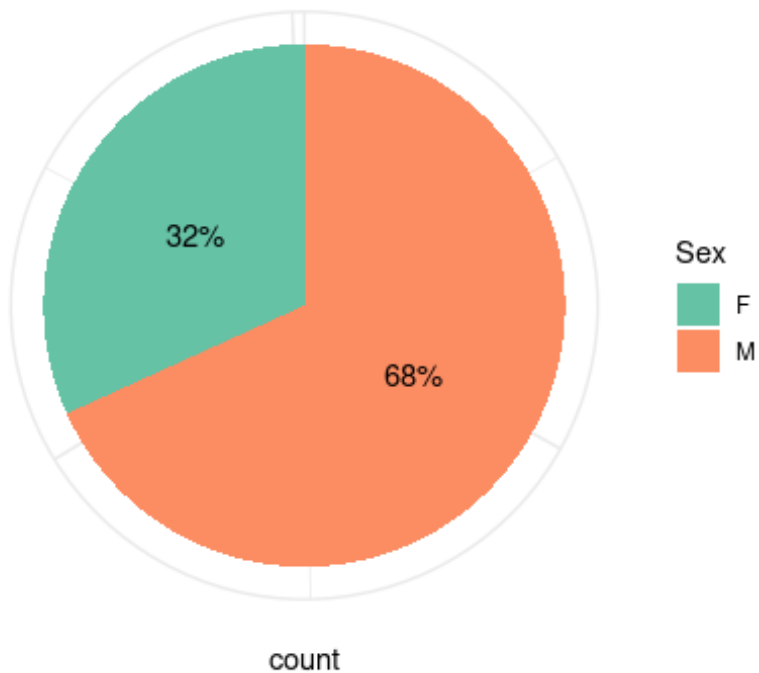
```
ggplot(data = df, aes(x = age, fill = output)) +  
  geom_histogram(bins = 30, color = "black" ) +  
  theme_minimal() +  
  labs(title = "Distribution of age") +  
  scale_fill_manual(values= c("#0cb02a", "#c71010"))+  
  xlab("Age (year old)")
```



Sex ratio

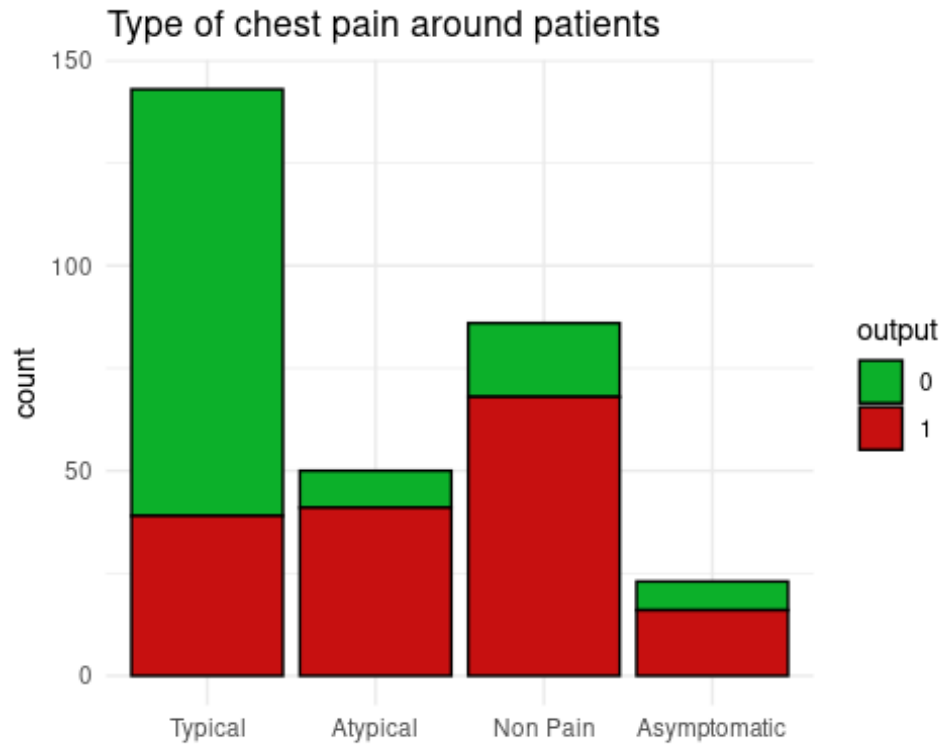
```
ggplot(df, aes(x = "", y = after_stat(count), fill = sex)) +  
  geom_bar(stat = "count", width = 1 ) +  
  coord_polar("y", start = 0) +  
  theme_minimal() +  
  labs(title = "Sex ratio", fill = "Sex", x = NULL ) +  
  scale_fill_brewer(palette = "Set2") +  
  theme(axis.text.x = element_blank()) +  
  geom_text(aes(label = paste0(round((after_stat(count) / sum(after_stat(count))) * 100), "%")), position = position_stack(vjust = 0.5), stat = "count")
```

Sex ratio



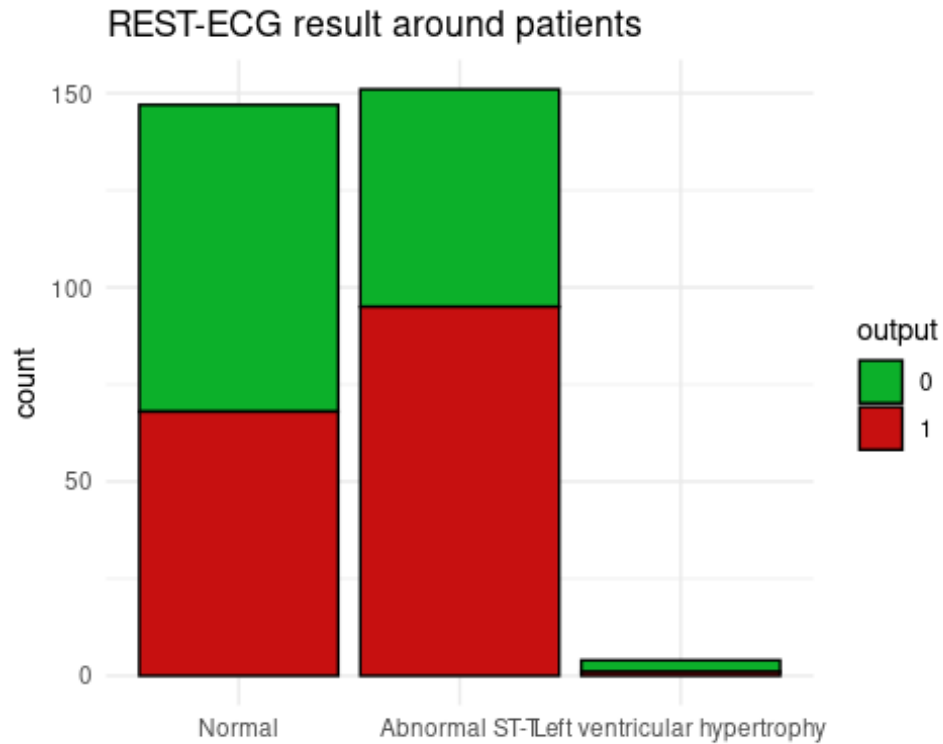
Type of chest pain around patients

```
ggplot(df, aes(x= cp, fill = output)) +  
  geom_bar(color = "black") +  
  theme_minimal() +  
  labs(title = "Type of chest pain around patients") +  
  scale_fill_manual(values= c("#0cb02a", "#c71010")) +  
  scale_x_discrete(labels = c("Typical", "Atypical", "Non Pain", "Asymptomatic")) +  
  xlab(element_blank())
```



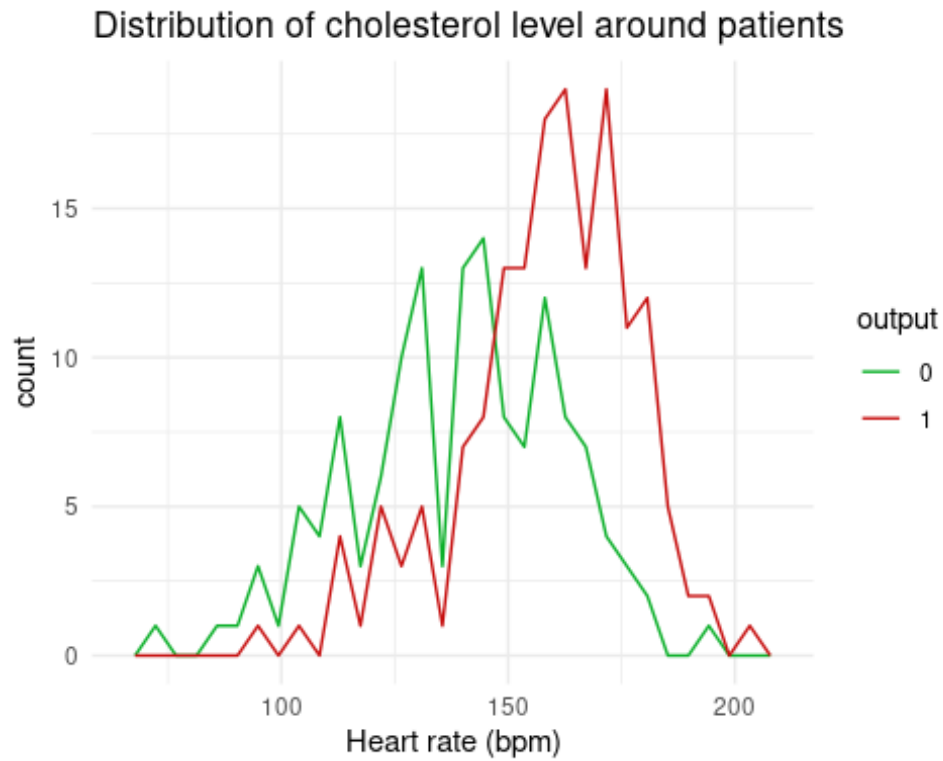
Resting electrocardiographic results around patients

```
ggplot(df, aes(x= restecg, fill = output)) +  
  geom_bar(color = "black") +  
  theme_minimal() +  
  labs(title = "REST-ECG result around patients") +  
  scale_fill_manual(values= c("#0cb02a", "#c71010")) +  
  scale_x_discrete(labels = c("Normal", "Abnormal ST-T", "Left ventricular hy  
pertrophy")) +  
  xlab(element_blank())
```



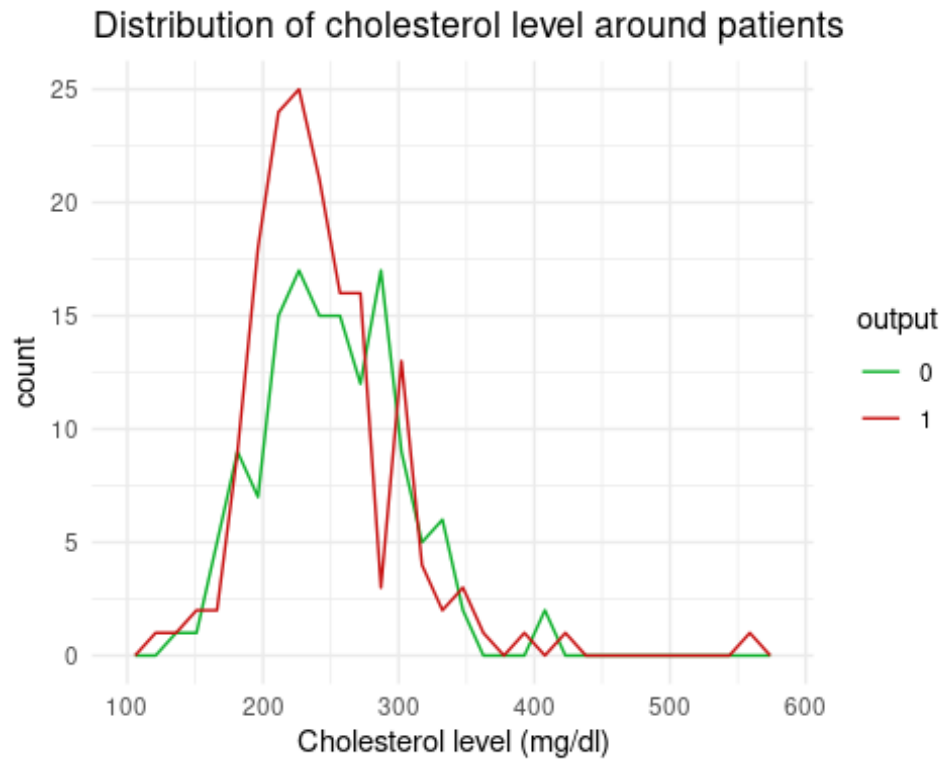
Heart rate around patients

```
ggplot(df, aes(x = thalachh , color = output)) +  
  geom_freqpoly() +  
  scale_color_manual(values= c("#0cb02a", "#c71010")) +  
  labs(title = "Distribution of cholesterol level around patients") +  
  xlab("Heart rate (bpm)") +  
  theme_minimal()
```



Cholesterol level around patients

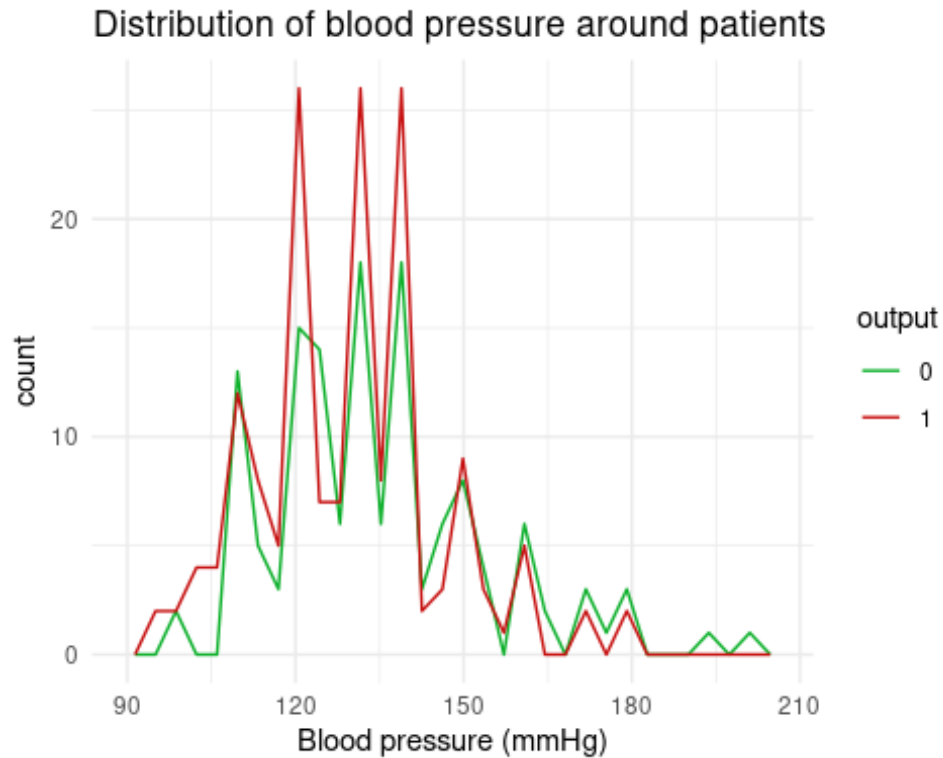
```
ggplot(df, aes(x = chol, color = output)) +  
  geom_freqpoly() +  
  scale_color_manual(values= c("#0cb02a", "#c71010")) +  
  labs(title = "Distribution of cholesterol level around patients") +  
  xlab("Cholesterol level (mg/dl)") +  
  theme_minimal()
```



Blood pressure level around patients

```
ggplot(df, aes(x = trtbps , color = output)) +  
  geom_freqpoly() +  
  scale_color_manual(values= c("#0cb02a", "#c71010")) +  
  labs(title = "Distribution of blood pressure around patients") +  
  xlab("Blood pressure (mmHg)") +  
  theme_minimal()
```

`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.



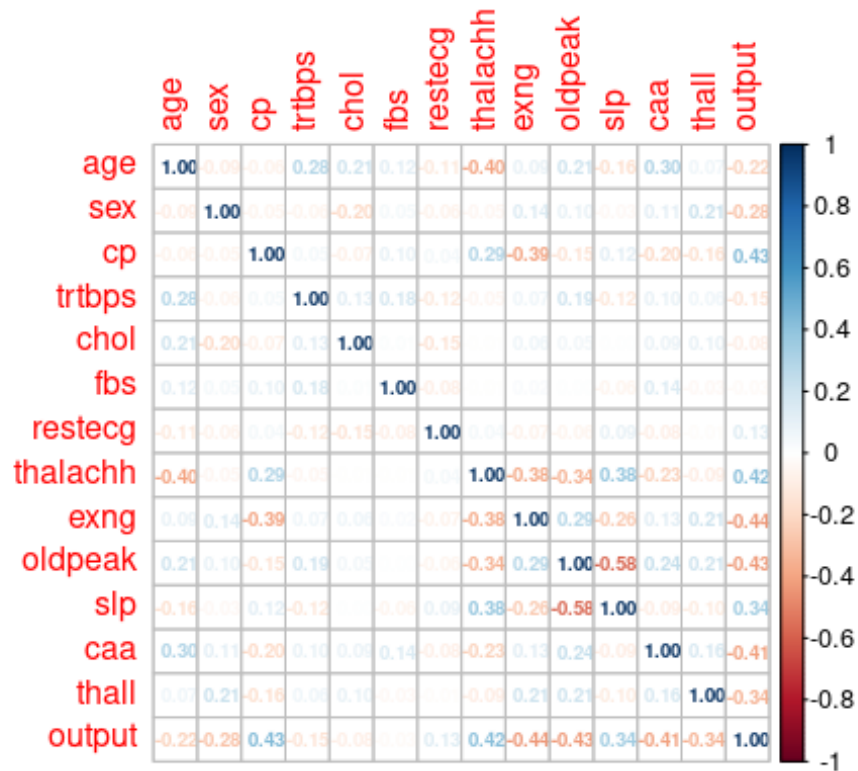
Correlation matrix

```
# Calculate the correlation matrix
```

```
corr_matrix <- cor(df)
```

```
# Create a correlation heat map
```

```
corrplot(corr_matrix, method= "number", number.cex = 0.6 )
```



Prediction model

Split data

```
split_data <- function(df, train_size = 0.8) {  
  set.seed(42)  
  n <- nrow(df)  
  id <- sample(1:n, size = n*train_size)  
  train_df <- df[id,]  
  test_df <- df[-id,]  
  return(list(train = train_df, test = test_df))  
}  
  
prep_data <- split_data(df)  
train_data <- prep_data[[1]]  
test_data <- prep_data[[2]]
```

Logistic regression

Train model

```
set.seed(25)  
ctrl <- trainControl(  
  method = "cv",  
  number = 5,  
  verboseIter = TRUE  
)  
  
lgt_model <- train(output ~ .,  
  data = train_data,  
  method = "glm",  
  preProcess = c("center", "scale"),  
  trControl = ctrl)
```

Model evaluation

```
# Train data  
p <- predict(lgt_model)  
confusionMatrix(p, train_data$output,  
  positive = "0",  
  mode = "prec_recall")  
  
## Confusion Matrix and Statistics  
##  
##           Reference  
## Prediction  0   1  
##           0  88 12  
##           1  21 120  
##  
##           Accuracy : 0.8631  
##           95% CI : (0.8131, 0.9038)  
##           No Information Rate : 0.5477  
##           P-Value [Acc > NIR] : <2e-16
```

```

##
##          Kappa : 0.7216
##
## Mcnemar's Test P-Value : 0.1637
##
##          Precision : 0.8800
##          Recall : 0.8073
##          F1 : 0.8421
##          Prevalence : 0.4523
##          Detection Rate : 0.3651
##          Detection Prevalence : 0.4149
##          Balanced Accuracy : 0.8582
##
##          'Positive' Class : 0
##

# Test data
p <- predict(lgt_model, newdata = test_data)
confusionMatrix(p, test_data$output,
                positive = "0",
                mode = "prec_recall")

## Confusion Matrix and Statistics
##
##          Reference
## Prediction  0  1
##          0 19  4
##          1 10 28
##
##          Accuracy : 0.7705
##          95% CI : (0.645, 0.8685)
##          No Information Rate : 0.5246
##          P-Value [Acc > NIR] : 6.666e-05
##
##          Kappa : 0.5354
##
## Mcnemar's Test P-Value : 0.1814
##
##          Precision : 0.8261
##          Recall : 0.6552
##          F1 : 0.7308
##          Prevalence : 0.4754
##          Detection Rate : 0.3115
##          Detection Prevalence : 0.3770
##          Balanced Accuracy : 0.7651
##
##          'Positive' Class : 0
##

```

Logistic regression with ridge and lasso

Train model

```
set.seed(25)
ctrl <- trainControl(
  method = "cv",
  number = 5,
  verboseIter = TRUE
)

my_grid <- expand.grid(alpha = 0:1,
  lambda = seq(0.0005, 0.05, length = 20))
glmnet_model <- train(output~.,
  data = train_data,
  method = "glmnet",
  preProcess = c("center", "scale"),
  tunGrid = my_grid,
  trControl = ctrl)
```

Model evaluation

```
# Train data
p <- predict(glmnet_model)
confusionMatrix(p, train_data$output,
  positive = "0",
  mode = "prec_recall")

## Confusion Matrix and Statistics
##
##              Reference
## Prediction    0    1
##           0  83  10
##           1  26 122
##
##              Accuracy : 0.8506
##              95% CI : (0.7992, 0.8931)
##      No Information Rate : 0.5477
##      P-Value [Acc > NIR] : < 2e-16
##
##              Kappa : 0.6946
##
##  Mcnemar's Test P-Value : 0.01242
##
##              Precision : 0.8925
##              Recall : 0.7615
##              F1 : 0.8218
##              Prevalence : 0.4523
##      Detection Rate : 0.3444
##      Detection Prevalence : 0.3859
##      Balanced Accuracy : 0.8429
##
```

```

##          'Positive' Class : 0
##

# Test data
p <- predict(glmnet_model, newdata = test_data)
confusionMatrix(p, test_data$output,
                 positive = "0",
                 mode = "prec_recall")

## Confusion Matrix and Statistics
##
##          Reference
## Prediction  0  1
##          0 19  1
##          1 10 31
##
##          Accuracy : 0.8197
##          95% CI : (0.7002, 0.9064)
##          No Information Rate : 0.5246
##          P-Value [Acc > NIR] : 1.492e-06
##
##          Kappa : 0.6331
##
##          Mcnemar's Test P-Value : 0.01586
##
##          Precision : 0.9500
##          Recall : 0.6552
##          F1 : 0.7755
##          Prevalence : 0.4754
##          Detection Rate : 0.3115
##          Detection Prevalence : 0.3279
##          Balanced Accuracy : 0.8120
##
##          'Positive' Class : 0
##

```

KNN model

Train model

```
set.seed(25)
ctrl <- trainControl(
  method = "cv",
  number = 5,
  verboseIter = TRUE
)

knn_model <- train(output ~ .,
  data = train_data,
  method = "knn",
  preProcess = c("center", "scale"),
  trControl = ctrl)
```

Model evaluation

```
# Train data
p <- predict(knn_model)
confusionMatrix(p, train_data$output,
  positive = "0",
  mode = "prec_recall")

## Confusion Matrix and Statistics
##
##              Reference
## Prediction    0    1
##              0  92  12
##              1  17 120
##
##              Accuracy : 0.8797
##              95% CI : (0.8318, 0.9179)
##      No Information Rate : 0.5477
##      P-Value [Acc > NIR] : <2e-16
##
##              Kappa : 0.7561
##
##  Mcnemar's Test P-Value : 0.4576
##
##              Precision : 0.8846
##              Recall : 0.8440
##              F1 : 0.8638
##              Prevalence : 0.4523
##              Detection Rate : 0.3817
##      Detection Prevalence : 0.4315
##              Balanced Accuracy : 0.8766
##
##              'Positive' Class : 0
##
```

```

# Test data
p <- predict(knn_model, newdata = test_data)
confusionMatrix(p, test_data$output,
                 positive = "0",
                 mode = "prec_recall")

## Confusion Matrix and Statistics
##
##              Reference
## Prediction  0  1
##           0 22  4
##           1  7 28
##
##              Accuracy : 0.8197
##              95% CI : (0.7002, 0.9064)
##      No Information Rate : 0.5246
##      P-Value [Acc > NIR] : 1.492e-06
##
##              Kappa : 0.6367
##
##  McNemar's Test P-Value : 0.5465
##
##              Precision : 0.8462
##              Recall : 0.7586
##              F1 : 0.8000
##              Prevalence : 0.4754
##              Detection Rate : 0.3607
##      Detection Prevalence : 0.4262
##      Balanced Accuracy : 0.8168
##
##      'Positive' Class : 0
##

```


Decision tree model

Train model

```
set.seed(25)
ctrl <- trainControl(
  method = "cv",
  number = 5,
  verboseIter = TRUE
)

rpart_model <- train(output~ .,
  data = train_data,
  method = "rpart",
  preProcess = c("center", "scale"),
  tuneGrid = expand.grid(cp = c(0.02, 0.0001, 0.25)),
  trControl = ctrl)
```

Model evaluation

```
# Train data
p <- predict(rpart_model)
confusionMatrix(p, train_data$output,
  positive = "0",
  mode = "prec_recall")

## Confusion Matrix and Statistics
##
##              Reference
## Prediction    0    1
##              0  89  14
##              1  20 118
##
##              Accuracy : 0.8589
##              95% CI : (0.8085, 0.9003)
##      No Information Rate : 0.5477
##      P-Value [Acc > NIR] : <2e-16
##
##              Kappa : 0.7139
##
##  McNemar's Test P-Value : 0.3912
##
##              Precision : 0.8641
##              Recall : 0.8165
##              F1 : 0.8396
##              Prevalence : 0.4523
##      Detection Rate : 0.3693
##      Detection Prevalence : 0.4274
##      Balanced Accuracy : 0.8552
##
##      'Positive' Class : 0
##
```

```

# Test data
p <- predict(rpart_model, newdata = test_data)
confusionMatrix(p, test_data$output,
                 positive = "0",
                 mode = "prec_recall")

## Confusion Matrix and Statistics
##
##              Reference
## Prediction  0   1
##           0 19   3
##           1 10  29
##
##              Accuracy : 0.7869
##              95% CI : (0.6632, 0.8814)
##      No Information Rate : 0.5246
##      P-Value [Acc > NIR] : 2.064e-05
##
##              Kappa : 0.5678
##
##  McNemar's Test P-Value : 0.09609
##
##              Precision : 0.8636
##              Recall : 0.6552
##              F1 : 0.7451
##              Prevalence : 0.4754
##              Detection Rate : 0.3115
##      Detection Prevalence : 0.3607
##      Balanced Accuracy : 0.7807
##
##      'Positive' Class : 0
##

```

Random forest model

Train model

```
set.seed(25)
ctrl <- trainControl(
  method = "cv",
  number = 5,
  verboseIter = TRUE
)

mtry_grid <- data.frame(mtry = 2:13)
rf_model <- train(output~ .,
  data = train_data,
  method = "rf",
  preProcess = c("center", "scale"),
  tuneGrid = mtry_grid,
  trControl = ctrl)
```

Model evaluation

```
# Train data
p <- predict(rf_model)
confusionMatrix(p, train_data$output,
  positive = "0",
  mode = "prec_recall")

## Confusion Matrix and Statistics
##
##              Reference
## Prediction    0    1
##              0 108    0
##              1    1 132
##
##              Accuracy : 0.9959
##              95% CI : (0.9771, 0.9999)
##      No Information Rate : 0.5477
##      P-Value [Acc > NIR] : <2e-16
##
##              Kappa : 0.9916
##
##  Mcnemar's Test P-Value : 1
##
##              Precision : 1.0000
##              Recall : 0.9908
##              F1 : 0.9954
##              Prevalence : 0.4523
##              Detection Rate : 0.4481
##              Detection Prevalence : 0.4481
##              Balanced Accuracy : 0.9954
##
```

```

##      'Positive' Class : 0
##

# Test data
p <- predict(rf_model, newdata = test_data)
confusionMatrix(p, test_data$output,
                 positive = "0",
                 mode = "prec_recall")

## Confusion Matrix and Statistics
##
##      Reference
## Prediction 0  1
##      0 21  2
##      1  8 30
##
##      Accuracy : 0.8361
##      95% CI : (0.7191, 0.9185)
##      No Information Rate : 0.5246
##      P-Value [Acc > NIR] : 3.442e-07
##
##      Kappa : 0.6681
##
##      Mcnemar's Test P-Value : 0.1138
##
##      Precision : 0.9130
##      Recall : 0.7241
##      F1 : 0.8077
##      Prevalence : 0.4754
##      Detection Rate : 0.3443
##      Detection Prevalence : 0.3770
##      Balanced Accuracy : 0.8308
##
##      'Positive' Class : 0
##

```

Model comparison

```
list_models = list(Lgt = lgt_model,
                   Glm = glmnet_model,
                   Knn = knn_model,
                   DecisionTree = rpart_model,
                   RandomForest = rf_model)
results = resamples(list_models)
summary(results)

##
## Call:
## summary.resamples(object = results)
##
## Models: Lgt, Glm, Knn, DecisionTree, RandomForest
## Number of resamples: 5
##
## Accuracy
##           Min.   1st Qu.   Median     Mean   3rd Qu.   Max. N
A's
## Lgt      0.7083333 0.7708333 0.8571429 0.8255952 0.8750000 0.9166667
0
## Glm      0.7291667 0.7708333 0.8775510 0.8421769 0.8958333 0.9375000
0
## Knn      0.7708333 0.7916667 0.8541667 0.8380102 0.8775510 0.8958333
0
## DecisionTree 0.6041667 0.6875000 0.7916667 0.7506803 0.8333333 0.8367347
0
## RandomForest 0.7291667 0.7500000 0.8163265 0.8132653 0.8750000 0.8958333
0
##
## Kappa
##           Min.   1st Qu.   Median     Mean   3rd Qu.   Max. N
A's
## Lgt      0.4000000 0.5368421 0.7100592 0.6448136 0.7464789 0.8306878
0
## Glm      0.4448399 0.5368421 0.7504244 0.6784865 0.7879859 0.8723404
0
## Knn      0.5335689 0.5833333 0.7052632 0.6728329 0.7525253 0.7894737
0
## DecisionTree 0.1943463 0.3856655 0.5862069 0.5014801 0.6683938 0.6727880
0
## RandomForest 0.4564460 0.4893617 0.6240409 0.6212392 0.7446809 0.7916667
0
```