MONASH University

A data mining technique to detect coronary artery disease using predictive modelling

---

Yupeng (Vita) Miao

Yunxuan Wang

Chang Yu Wang

Monash University

Faculty of Information Technology

Code and Test Report for
FIT3164 Group Project

Full source code can be found at
`https://github.com/cywng/monash-datascience-proj-heartdisease`

October 2020

Contents

2 Code report: Applications

This continues from Code report: Models.

2.1 Source Code

2.1.1 API

The API comprises of two files, `plumber_api.R` and `svm_api.R`. The file `plumber_api.R` sets up the backend for the API, taking inputs (lines 27-43), processing them, and then returning a result via function `calculate_prediction()`. The file `svm_api.R` takes this file and hosts it locally on port 8000.

plumber_api.R

```
1  #Script name: plumber_api.R
2
3  #' @apiTitle Run predictions for Coronary Artery Disease with a SVM model.
4  #' @apiDescription This API takes as patient data on Coronary artery disease
       (CAD) and returns the probability of having CAD (between 0 and 1).
5  #' This was based on the web post https://www.shirin-glander.de/2018/01/
       plumber/.
6  #' This has been built as part of a final year project at Monash university,
       with repo being available here:
7  #' https://github.com/cywng/monash-datascience-proj-heartdisease.
8
9  # load model
10 load(file="../Models/svm_linear.RData")
11
12 #' Log system time, request method and HTTP user agent of the incoming
       request
13 #' @filter logger
14 function(req){
15   cat("System time:", as.character(Sys.time()), "\n",
16       "Request method:", req$REQUEST_METHOD, req$PATH_INFO, "\n",
17       "HTTP user agent:", req$HTTP_USER_AGENT, "@", req$REMOTE_ADDR, "\n")
18   plumber::forward()
19 }
20
21 # core function follows below:
22 # define parameters with type and description
23 # name endpoint
24 # return output as html/text
25 # specify 200 (okay) return
26
27 #' predict Coronary Artery Disease with an SVM model.
28 #' @param Typical.Chest.Pain:numeric Presence of chest pain (1 for yes, 0
       for no).
29 #' @param Age:numeric The age of the patient.
30 #' @param Atypical:int Atypical pulse (1 for yes, 0 for no).
```

```
#' @param FBS:numeric Fasting blood sugar level (mg/dl, 62-400).
#' @param HTN:int Presence of hypertension (1 for yes, 0 for no).
#' @param DM:int Presence of diabetes mellitus (1 for yes, 0 for no).
#' @param EF.TTE:numeric Level of Ejection Fraction (15-60).
#' @param K:numeric Potassium content in blood (mEq/lit; 3.0-6.6).
#' @param ESR:numeric Erythrocyte sedimentation rate (mm/h; 1-90).
#' @param TG:numeric Triclyceride concentration in blood (mg/dl; 37-1050).
#' @param Region.RWMA:int If any region of the heart has regional wall
    motion abnormality (0-4, 0 for none).
#' @param BP:numeric Blood Pressure (mmHgl 90-190).

#' @get /predict
#' @html
#' @response 200 Returns the class (Y or N) prediction from the LDA model; Y
     = Coronary Artery Disease
calculate_prediction <- function(Typical.Chest.Pain, Age, Atypical, FBS, HTN
    , DM, EF.TTE, K, ESR, TG, Region.RWMA, BP) {
  Atypical <- ifelse(Atypical==1,"Y","N")
  #Makes a prediction on the input parameters using the SVM model.

  # make data frame from parameters
  input_data_num <<- data.frame(Age, FBS, EF.TTE, K, ESR, TG, BP,
      stringsAsFactors = FALSE)
  # and make sure they really are numeric
  input_data_num <<- as.data.frame(t(sapply(input_data_num, as.numeric)))

  # make data frame from parameters
  input_data_fac <<- data.frame(Typical.Chest.Pain, Atypical, HTN, DM,
      Region.RWMA)#, stringsAsFactors = FALSE)
  # convert to factor
  input_data_fac <<- as.data.frame(t(lapply(input_data_fac, as.factor)))
  #Combine into one df
  input_data <<- as.data.frame(c(input_data_num, input_data_fac))
  paste(input_data)
  # validation for parameter
  if (any(is.na(input_data))) {
    res$status <- 400
    res$body <- "Parameters have to be numeric or integers"
  }

  # predict and return result
  pred_svm <- round(predict(svm.l,input_data,type = "prob")$Cad,2)

  paste("----------------\nThe case has a ",as.character(pred_svm)," chance
      of having Coronary Artery Disease.\n----------------\n")
}
```

svm_api.R

```
library(plumber)
```

```
r <- plumb("./API/plumber_api.R")
r$run(port = 8000)

# we can access it via *http://localhost:8000/predict*, but requires an
    input
# in a JSON format. Examples can be found in the github repo.
```

2.1.2 Website

The website is also built from two files, `ui.R` and `server.R`. Shiny uses `ui.R` to build the user interface and get inputs, while using `server.R` to do the input sanitation and prediction.

ui.R

```
library(shiny)
library(caret)
library(kernlab)
load(file = "svm_linear.RData")
load(file = "svm_train_data.RData")

Age<-round(mean(svm.train$Age),digits=0)  #
DM<-as.factor(round(mean(as.numeric(svm.train$DM)),digits=0))#
HTN<-as.factor(round(mean(as.numeric(svm.train$HTN)),digits=0))#
BP<-round(mean(svm.train$BP),digits=0)
Typical.Chest.Pain<-as.factor(round(mean(as.numeric(svm.train$Typical.Chest.
    Pain)),digits=0))
Atypical<-as.factor(round(mean(as.numeric(svm.train$Atypical)),digits=0))
FBS<-round(mean(svm.train$FBS),digits=0)#
TG<-round(mean(svm.train$TG),digits=0)#
ESR<-round(mean(svm.train$ESR),digits=0)#
K<-round(mean(svm.train$K),digits=0)#
EF.TTE<-round(mean(svm.train$EF.TTE),digits=0)#
Region.RWMA<-as.factor(round(mean(as.numeric(svm.train$Region.RWMA)),digits
    =0))

fluidPage(
  h1("Heart Disease Prediction",align = "center"),#titlePanel
  p("This is a tool to aid the diagnosis of Coronary Artery Disease. Please
     input the patient's information into the fields, or upload a CSV."),
  p("Typical values have been displayed brackets. An entry of 0 for a
     numeric category is treated as a null input. Blank entries or entries
     outside the range are acceptable, but will result in a lower confidence
      than displayed."),

  br(),

  sidebarLayout(
    sidebarPanel (
      fluidRow(
```

```
30          #
31          column(width =6 ,selectInput("Typical.Chest.Pain","Presence of chest
                pain:",c("Unknown","Yes","No"),multiple = F,selected =NULL)),
32        #
33            column(width =6 ,numericInput("Age","Age (30-86):",0))),
34
35        fluidRow(
36          #
37          column(width =6, selectInput("Atypical","Atypical Pulse:",c("Unknown
              ","Yes","No"),multiple = F,selected =NULL)), #N,Y
38         #
39           column(width =6,numericInput("FBS","Fasting blood sugar level (mg/
                dl; 62-400):",0))),
40
41        fluidRow(
42          #
43          column(width =6,selectInput("HTN","Hypertension:",c("Unknown","Yes",
              "No"),multiple = F,selected =NULL)),
44          #
45          column(width =6,selectInput("DM","Diabetes Mellitus:",c("Unknown","
              Yes","No"),multiple = F,selected =NULL))),
46
47        fluidRow(
48          #
49          column(width =6,numericInput("EF.TTE","Ejection Fraction (%; 15-60):
              ",0)),
50          #
51          column(width =6,numericInput("K","Blood potassium Content (mEq/lit;
              3.0-6.6):",0))), #float
52
53        fluidRow(
54          #
55          column(width =6,numericInput("BP","Blood pressure (mmHg; 90  190  ):"
              ,0)),
56          #
57          column(width =6,numericInput("ESR","Erythrocyte sedimentation rate (
              mm/h; 1-90):",0))),
58
59        fluidRow(
60          #
61          column(width =6,numericInput("TG","Blood triclyceride concentration
              (mg/dl; 35-1050):",0)),
62          #need modify
63          column(width =6,selectInput("Region.RWMA","Regional wall motion
              abnormality in heart region:",c(0,1,2,3,4),multiple = F,selected
              =NULL))),#region,RWAMA:023,
64
65
66        fileInput("file1","or upload a CSV File",accept = ".csv"),
```

```
67
68        actionButton("button", "Get prediction",width = "100%",style="color: #
              fff; background-color: #337ab7; border-color: #2e6da4"),
69        #  submitButton("Submit",width = "100%"),
70        width="100%"),
71      mainPanel(
72        textOutput("pred" ),
73        tags$head(tags$style("#pred{color: black;
74                                    font-size: 20px;
75                                    }"
76        )
77        )
78
79      )
80    )
81  )
```

server.R

```
1  library(shiny)
2  library(Hmisc)
3  library(kernlab)
4  library
5  load(file = "svm_linear.RData")
6  load(file = "svm_train_data.RData")
7
8
9
10 #to test changes locally set wd to folder containing server.R and ui.R:
       library(shiny); runapp()
11 #to push changes: library(rsconnect); deployApp()
12
13 function(input, output) {
14   observeEvent(input$button,{
15     output$pred<-renderPrint({
16       #If no input file
17       if(is.null(input$file1)){
18
19         #Set values to train data averages if no input.
20         if(input$Typical.Chest.Pain=="Unknown"){Typical.Chest.Pain<-as.
               factor(round(mean(as.numeric(svm.train$Typical.Chest.Pain)-1),
               digits=0))}
21         else{Typical.Chest.Pain<-as.factor(as.numeric(ifelse(input$Typical.
               Chest.Pain=="Yes",1,0)))}#
22
23
24         if(input$Age==0 || is.na(input$Age)){Age<-round(mean(svm.train$Age),
               digits=0)}
25         else{Age<-input$Age}#
26
```

```
27          if(input$Atypical=="Unknown"){Atypical<-"N"}
28          else{Atypical<-as.factor( ifelse(input$Atypical=="Yes","Y","N"))} #
29
30          if(input$FBS==0 || is.na(input$FBS)){FBS<-round(mean(svm.train$FBS),
                digits=0)}
31          else{FBS<-input$FBS} #
32
33          if(input$HTN=="Unknown"){HTN<-as.factor(round(mean(as.numeric(svm.
                train$HTN)-1),digits=0))}
34          else{HTN<-as.factor(as.numeric(ifelse(input$HTN=="Yes",1,0)))} #
35
36          if(input$DM=="Unknown"){DM<-as.factor(round(mean(as.numeric(svm.
                train$DM)-1),digits=0))}
37          else{DM<-as.factor(as.numeric(ifelse(input$DM=="Yes",1,0)))} #
38
39          if(input$EF.TTE==0 || is.na(input$EF.TTE)){EF.TTE<-round(mean(svm.
                train$EF.TTE),digits=0)}
40          else{EF.TTE<-input$EF.TTE} #
41
42          if(input$K==0 || is.na(input$K)){K<-round(mean(svm.train$K),digits
                =0)}
43          else{K<-input$K} #
44
45          if(input$BP==0 || is.na(input$BP)){BP<-round(mean(svm.train$BP),
                digits=0)}
46          else{BP<-input$BP}  #
47
48          if(input$ESR==0 || is.na(input$ESR)){ESR<-round(mean(svm.train$ESR),
                digits=0)}
49          else{ESR<-input$ESR}  #
50
51          if(input$TG==0 || is.na(input$TG)){TG<-round(mean(svm.train$TG),
                digits=0)}
52          else{TG<-input$TG} #
53
54          if(input$Region.RWMA=="Unknown"){Region.RWMA<-as.factor(round(mean(
                as.numeric(svm.train$Region.RWMA)-1),digits=0))}
55          else{Region.RWMA<-as.factor(input$Region.RWMA)}
56
57
58          new<-data.frame("Typical.Chest.Pain"=as.factor(Typical.Chest.Pain),"
                Age"=Age,
59                          "Atypical"=as.factor(Atypical),"FBS"=FBS,"HTN"=as.
                               factor(HTN),
60                          "DM"=as.factor(DM),"EF.TTE"=EF.TTE, "K"=K,"BP"=BP,"
                               ESR"=ESR,
61                          "TG"=TG,"Region.RWMA"=as.factor(Region.RWMA))
62          pr=round(predict(svm.l,new[,!names(new) %in% c("Cath")],type = "prob
                ")$Cad,2)
```

```
63          #predict(svm.l,svm.test[,!names(svm.test) %in% c("Cath")],type = "
                prob")$Cad
64          #pr=round(predict(svm.l,new,type="prob")$Y,2)
65          cat("The case has a ",pr," chance of having Coronary Artery Disease.
                ")
66          #paste0(pr)
67
68        }
69        #This is for if there is an input file. Check for no entry with is.na
              ()
70        else{
71          file<-input$file1
72
73          ext <- tools::file_ext(file$datapath)
74          req(file)
75          validate(need(ext == "csv", "Please upload a csv file"))
76          file<-read.csv(file$datapath)
77
78          if(is.na(file$Typical.Chest.Pain) || is.null(file$Typical.Chest.Pain
                ) ){Typical.Chest.Pain<-as.factor(round(mean(as.numeric(svm.train
                $Typical.Chest.Pain)-1),digits=0))}
79          else{as.factor(Typical.Chest.Pain<-file$Typical.Chest.Pain)}#
80
81          if(is.na(file$Age) || is.null(file$Age) ){Age<-round(mean(svm.train$
                Age),digits=0)}
82          else{Age<-file$Age} #
83
84          if(is.na(file$Atypical) || is.null(file$Atypical) ){Atypical<-"N"}
85          else{Atypical<-as.factor(file$Atypical)} #
86
87          if(is.na(file$FBS) || is.null(file$FBS) ){FBS<-round(mean(svm.train$
                FBS),digits=0)}
88          else{FBS<-file$FBS} #
89
90          if(is.na(file$HTN) || is.null(file$HTN) ){HTN<-as.factor(round(mean(
                as.numeric(svm.train$HTN)-1),digits=0))}
91          else{HTN<-as.factor(as.numeric(file$HTN))} #
92
93          if(is.na(file$DM)  || is.null(file$DM)){DM<-as.factor(round(mean(as.
                numeric(svm.train$DM)-1),digits=0))}
94          else{DM<-as.factor(as.numeric(file$DM))} #
95
96          if(is.na(file$EF.TTE) || is.null(file$EF.TTE)){EF.TTE<-round(mean(
                svm.train$EF.TTE),digits=0)}
97          else{EF.TTE<-file$EF.TTE}#
98
99          if(is.na(file$K)|| is.null(file$K)){K<-round(mean(svm.train$K),
                digits=0)}
100         else{K<-file$K} #
```

```
101
102         if(is.na(file$BP) || is.null(file$BP)){BP<-round(mean(svm.train$BP),
                digits=0)}
103         else{BP<-file$BP} #
104
105         if(is.na(file$ESR) || is.null(file$ESR)){ESR<-round(mean(svm.train$
                ESR),digits=0)}
106         else{ESR<-file$ESR}  #
107
108         if(is.na(file$TG)|| is.null(file$TG)){TG<-round(mean(svm.train$TG),
                digits=0)}
109         else{TG<-file$TG} #
110
111
112         if(is.na(file$Region.RWMA)|| is.null(file$Region.RWMA)){Region.RWMA
                <-as.factor(round(mean(as.numeric(svm.train$Region.RWMA)-1),
                digits=0))}
113         else{Region.RWMA<-as.factor(file$Region.RWMA)}
114
115
116
117         #-------
118         new2<-data.frame("Typical.Chest.Pain"=as.factor(Typical.Chest.Pain),
                "Age"=Age,
119                          "Atypical"=as.factor(Atypical),"FBS"=FBS,"HTN"=as.
                             factor(HTN),
120                          "DM"=as.factor(DM),"EF.TTE"=EF.TTE, "K"=K,"BP"=BP,"
                             ESR"=ESR,
121                          "TG"=TG,"Region.RWMA"=as.factor(Region.RWMA))
122         pr=round(predict(svm.l,new2[,!names(new2) %in% c("Cath")],type = "
                prob")$Cad,2)
123       #  pr=round(predict(svm.l,new2,type="prob")$Y,2)
124         # if(pr=="N"){cat("The reault is: NORMAL")}else if (pr=="Yes"){cat("
                The reault is: CAD")}
125         cat("The case has a ",pr," chance of having Coronary Artery Disease.
                ")
126         #paste0(pr)
127       }
128
129     })
130   })
131
132 }
```

2.2 Code Limitations

We have achieved the key deliverables laid out in the project proposal last semester. However, due to time, study and other constraints, there are additional features that have been

implemented.

2.2.1 Domain Limitations

As none of our group members have any domain specific knowledge, we could not tailor the models to suit the data. For example, if the project was on natural language processing, our knowledge of language would indicate that we should use some form of recurrent neural network, as sentences can be thought of as time-series data.

In this project, we had no deep understanding of heart disease and therefore could not draw on background info to guide our analysis. Hence, we blindly applied standard techniques on the data without any customisation, instead of building custom neural network architecture or kernels for SVM. Future projects could consult with an expert who would advice on ways to transform parameters or consider the data in line with current medical understanding.

2.2.2 Technical Limitations

Despite our best efforts, our models plateaued at around 88% test accuracy, with few making it to 90%. Even our attempts at ensembling strong models did not have much effect. We hypothesise that this is due to our simple applications of the models being unable to caputre the complex structure of the problem. Custom made machine learning models would likely perform better in this regard.

Due to a lack of time, our website and API are also quite simple. The API is local-host only, but could be imaged and uploaded in future releases. The app could also be improved to give more insight, such as making recommendations on methods to reduce the patient's risk of heart disease, or displaying graphics to show where the user stands in comparison to the average Australian.

2.3 End User Guide

2.3.1 Website Quick Start

Visit the website here `https://cywng.shinyapps.io/fit3164team1cadprediction/`.

1. Enter details about the patient into the boxes provided.
    - Leaving variables as 'Unknown' or 0 will use default values, reducing the accuracy of prediction.
    - Each numeric feature has a suggested range, but can accept inputs outside the range. Note that accuracy is lower outside this range, sometimes significantly.
2. Alternatively, upload a CSV file. The .csv should have one entry, with appropriate feature names. An example can be found here: `https://raw.githubusercontent.com/cywng/monash-datascience-proj-heartdisease/master/cad.csv`
3. Click "Get prediction" at the bottom of the page.

2.3.2 Warnings

Any results obtained from the model are not a substitute for professional medical advice. The tool is designed to be used in conjunction with expert knowledge from a medical professional, not as the sole method of diagnosis.

Furthermore, CAD affects 2.8% of the adult population in Australia. As our testing accuracy has been 90%, there is a high chance of producing a false positive when selecting a random person from the population. Hence, this should be used for patients in an at-risk category, and supplemented with further testing.

2.3.3 Github Repo

The full source code can be found here:

`https://github.com/cywng/monash-datascience-proj-heartdisease`

See Technical User Guide for more information.

2.4 Technical User Guide

This code is written in R using numerous packages, which will need to be installed. Notable packages include *caret, ggplot2, xlsx, pROC, plyr, e1071, shiny, kernlab, plumber* and their dependencies. Download the source code here:

`https://github.com/cywng/monash-datascience-proj-heartdisease`.

2.4.1 Website testing and deployment

Open the source code in R studio and set working directory to the website folder as shown below. In R studio, this can be done as follows:

1. Session → Set working directory → Choose directory.
2. Navigate to
   `./monash-datascience-proj-heartdisease/website/FIT3164team1cadprediction`
   and select 'open'.

To test any changes to the app, type the following into console.

```
library(shiny)
runApp()
```

To make changes to the app, alter the files 'ui.R' and 'server.R' for the UI and back end respectively. Finally, to deploy the changes, run the following.

```
library(rsconnect)
deolpyApp()
```

This will update the website `https://cywng.shinyapps.io/fit3164team1cadprediction/`.

2.4.2 Local API deployment

Open the source code in R studio and set working directory to the base folder as shown below. In R studio, this can be done as follows:

1. Session → Set working directory → Choose directory.
2. Navigate to `./monash-datascience-proj-heartdisease/` and select open.

Navigate to `./API/` and run the 'lda_api.R' script. This will host the API locally and launch a swagger app that allows for ease of testing. While the swagger app is open, navigate to terminal (located next to console in Rstudio) to test API requests.

Test cases have been prepared as comments in the 'lda_api.R' script for convenience.

Finally, port forward port 8000 to launch the API. Alternatively, a docker image can be made on linux or through Docker Desktop on windows.

2.4.3 Adapting models

To get an understanding of the models, we recommend reading through CODE DEMO.PDF. Existing models can be found in `./Models (withfeature selection)/` and can be loaded and freely used.

- Models can be altered by navigating to `./Code demo.Rmd` and retraining the models using new data and manually saving outputs. They can also be found in `./Model Training/`, which saves the model to `./Models`.
- The ensemble functions are quite flexible, and additional models can be ensembled by altering `generate_ensemble_df()` in `Code demo.Rmd` or `./Ensemble/Custom_ensemble_prob.R`.
- The API and Website can run with different models by loading the required file into `./API/plumber_api.R` and `./website/FIT3164team1cadprediction/server.R` respectively. To do this:
  1. Alter the file loaded in `load(file=''path'')`.
  2. Search for all instances of `predict()` and update the model used. Note that to deploy the website, all required files, including models, should be in the `./website/FIT3164team1cadprediction` folder.

3 Test Report

3.1 Introduction

The testing in the project was to ensure the software satisfied the requirements, as specified in the initial proposal. Our models were to be as accurate as we could get, with a working

API and website to allow others to access our findings. Tests were conducted through both automatically and manually.

3.2 Preliminaries

3.2.1 Test: Data wrangling output

Manual test. This example is an exert from the entire data wrangling process, as each step was done manually, and hence was manually tested. Evidence of tests can be found in section **1.0** as the structure is inspected at each step.

a. What: That our code to convert characters to factors works.

b. How: Manual inspection of result, through use of `str(cad)`

c. Expected: A summary of the feature parameters, all of which being either 'num' or 'Factor'.

d. Result: As expected.

3.3 Model Tests

We tested the performance of all models trained, so that we could make an informed selection of best model to present in our API and website. This is inline with the prediction accuracy requirement. The SVM_linear model was chosen.

3.3.1 Test: Performance of Individual models

Here we use `caret`'s `confusionMatrix` function. This compares the predicted outcome to the actual outcome in the testing data set. We also inspect the ROC during training. This code has been extracted from the source code provided earlier.

a. Expected: Classification accuracy significantly better than 0.5 but not 1, as this would indicate random guessing or overfitting, respectively.

b. Results:

```
# Test SVM polynomial
svm.poly.predict = predict(svm.poly, svm.test)
# confusion matrix
cm.svm.poly <- confusionMatrix(svm.poly.predict, svm.test$Cath)
print(cm.svm.poly)
```

```
## Confusion Matrix and Statistics
##
## Reference
```

```
## Prediction Cad Normal
## Cad 40 5
## Normal 3 12
##
## Accuracy : 0.8667
## 95% CI : (0.7541, 0.9406)
```

```r
# Test SVM Radial
svm.rad.predict = predict(svm.rad, svm.test)
# confusion matrix
cm.svm.rad <- confusionMatrix(svm.rad.predict, svm.test$Cath)
print(cm.svm.rad)
```

```
## Confusion Matrix and Statistics
##
## Reference
## Prediction Cad Normal
## Cad 40 7
## Normal 3 10
##
## Accuracy : 0.8333
## 95% CI : (0.7148, 0.9171)
```

```r
# Test SVM linear
svm.l.predict = predict(svm.l, svm.test)
# confusion matrix
cm.svm.l <- confusionMatrix(svm.l.predict, svm.test$Cath)
print(cm.svm.l)
```

```
## Confusion Matrix and Statistics
##
## Reference
## Prediction Cad Normal
## Cad 42 5
## Normal 1 12
##
14
## Accuracy : 0.9
## 95% CI : (0.7949, 0.9624)
```

```r
# Test the random forest
rf.predict = predict(rf, svm.test)
# confusion matrix
cm.rf <- confusionMatrix(rf.predict, svm.test$Cath)
print(cm.rf)
```

```
## Confusion Matrix and Statistics
##
## Reference
## Prediction Cad Normal
## Cad 40 5
## Normal 3 12
##
## Accuracy : 0.8667
## 95% CI : (0.7541, 0.9406)
```

```r
# testing the neural network
nn.predict = predict(nn, svm.test)
# confusion matrix
cm.nn <- confusionMatrix(nn.predict, svm.test$Cath)
print(cm.nn)
```

```
## Confusion Matrix and Statistics
##
## Reference
## Prediction Cad Normal
## Cad 37 4
## Normal 6 13
##
## Accuracy : 0.8333
## 95% CI : (0.7148, 0.9171)
```

```r
# Test the Naive Bayes
nb.predict = predict(nb, svm.test)
# confusion matrix
cm.nb <- confusionMatrix(nb.predict, svm.test$Cath)
print(cm.nb)
```

```
## Confusion Matrix and Statistics
##
## Reference
## Prediction Cad Normal
## Cad 24 1
## Normal 19 16
##
## Accuracy : 0.6667
## 95% CI : (0.5331, 0.7831)
```

```r
# Test the logistic regression
lr.predict = predict(lr, svm.test)
# confusion matrix
cm.lr <- confusionMatrix(lr.predict, svm.test$Cath)
print(cm.lr)
```

```
## Confusion Matrix and Statistics
##
## Reference
## Prediction Cad Normal
## Cad 39 4
## Normal 4 13
##
## Accuracy : 0.8667
## 95% CI : (0.7541, 0.9406)
```

```r
# Test the lda
lda.predict = predict(lda, svm.test)
# confusion matrix
cm.lda <- confusionMatrix(lda.predict, svm.test$Cath)
print(cm.lda)
```

```
## Confusion Matrix and Statistics
##
## Reference
## Prediction Cad Normal
## Cad 39 4
## Normal 4 13
##
```

```
## Accuracy : 0.8667
## 95% CI : (0.7541, 0.9406)
```

```r
# Test the knn
knn.predict = predict(knn, svm.test)
# confusion matrix
cm.knn <- confusionMatrix(knn.predict, svm.test$Cath)
print(cm.knn)
```

```
## Confusion Matrix and Statistics
##
## Reference
## Prediction Cad Normal
## Cad 39 6
## Normal 4 11
##
## Accuracy : 0.8333
## 95% CI : (0.7148, 0.9171)
```

```r
# Test the decision tree
dt.predict = predict(dt, svm.test)
# confusion matrix
cm.dt <- confusionMatrix(dt.predict, svm.test$Cath)
print(cm.dt)
```

```
## Confusion Matrix and Statistics
##
## Reference
## Prediction Cad Normal
## Cad 39 3
## Normal 4 14
##
## Accuracy : 0.8833
## 95% CI : (0.7743, 0.9518)
```

3.3.2 Test: Performance of Ensemble models

We do perform the same tests as above, but on a collection of ensemble methods, including custom functions.

a. Expected: Classification accuracy significantly better than 0.5 but not 1, as this would indicate random guessing or overfitting, respectively.

b. Results:

```
# Test the gbm
gbm.predict = predict(gbm, svm.test)
# confusion matrix
cm.gbm <- confusionMatrix(gbm.predict, svm.test$Cath)
print(cm.gbm)
```

```
## Confusion Matrix and Statistics
##
27
## Reference
## Prediction Cad Normal
## Cad 38 4
## Normal 5 13
##
## Accuracy : 0.85
## 95% CI : (0.7343, 0.929)
```

```
# Testing AdaBoost Classification tree
adab.tree.predict = predict(adaB, svm.test)
# confusion matrix
cm.adab.tree <- confusionMatrix(adab.tree.predict, svm.test$Cath)
print(cm.adab.tree)
```

```
## Confusion Matrix and Statistics
##
## Reference
## Prediction Cad Normal
## Cad 38 4
## Normal 5 13
##
## Accuracy : 0.85
## 95% CI : (0.7343, 0.929)
```

```
# Testing Boosted logistic regression
blr.predict = predict(blr, svm.test)
# confusion matrix
cm.blr <- confusionMatrix(blr.predict, svm.test$Cath)
print(cm.blr)
```

```
## Confusion Matrix and Statistics
##
## Reference
## Prediction Cad Normal
## Cad 34 4
## Normal 5 11
##
## Accuracy : 0.8333
## 95% CI : (0.7071, 0.9208)
```

```
#Custom function: vote ensemble
ensem_result <- vote_ensemble(generate_ensemble_df(svm.train))
confusionMatrix(ensem_result,svm.train$Cath)
```

```
## Confusion Matrix and Statistics
##
## Reference
## Prediction Cad Normal
## Cad 163 8
## Normal 10 62
##
## Accuracy : 0.9259
## 95% CI : (0.8855, 0.9555)
```

```
#Custom function: logistic regression final layer ontop of ensembled
   data frame.
ensem_lr_test=predict(lr_ensem, ensem_test)
cm.lr_ensem = confusionMatrix(ensem_lr_test, svm.test$Cath)
cm.lr_ensem
```

```
## Confusion Matrix and Statistics
##
## Reference
## Prediction Cad Normal
## Cad 39 3
## Normal 4 14
##
## Accuracy : 0.8833
## 95% CI : (0.7743, 0.9518)
```

3.4 Website and API tests

3.4.1 Test: API hosted

Manual test:

a. What: That the API launches when running `svn_api.R`.

b. How: Running the file and attempting to get a request.

c. Expected: A message confirming the running of the API.

d. Result: As expected.

```
> library(plumber)
Warning message:
In load(path, envir = .GlobalEnv) :
  strings not representable in native encoding will be translated to
     UTF-8
> r <- plumb("./API/plumber_api.R")
Warning message:
Plumber tag ‘#* @html‘ is deprecated.
Use ‘#* @serializer html‘ instead.
> r$run(port = 8000)
Running plumber API at http://127.0.0.1:8000
Running swagger Docs at http://127.0.0.1:8000/__docs__/
```

3.4.2 Test: API returns same as prediction

Manual test:

a. What: The api delivers the correct prediction when requested through the terminal.

b. How: Sending a request while the terminal API is running. An example is attached.

```
$    curl -H "Content-Type: application/json" -X GET -d '{"Age":58,"DM
   ":"0","HTN":"0","BP":90,"Typical.Chest.Pain":"0","Atypical":"N","FBS
   ":69,"TG":79,"ESR":5,"K":3.4,"EF.TTE":50,"Region.RWMA":"0"}' "http:/
   /localhost:8000/predict"
```

c. Expected: A prediction that is identical to the model's prediction.

d. Result: As expected, as this was what the model had predicted.

```
----------------
The case has a  0.3  chance of having Coronary Artery Disease.
----------------
```

3.4.3 Test: Website prediction unit test

We use a testing framework called `testthat` for our unit testing, to test whether the predictions given by the website are the same as the result given by the selected model. We write the code inside `testthat` function: select the first 10 records of the training dataset as the inputs to the website and then check if the output of the website is the same as the output of the model.

Unit testing code: **OutputTest.R**

```
library(stringr)
library(shinytest)
library(testthat)
library(here)
load((here(file ="website/FIT3164team1cadprediction/svm_linear.Rdata"))
    )
load((here(file ="website/FIT3164team1cadprediction/svm_train_data.
    Rdata")))
context("Test Shiny app")
app <- shinytest::ShinyDriver$new("~/monash-datascience-proj-
    heartdisease/website/FIT3164team1cadprediction/")
test_that("output is correct", {
  i=1
  while (i<=10){
    if(as.numeric(svm.train$Typical.Chest.Pain[i])==0){ Typical.Chest.
        Pain<-"No"}else{ Typical.Chest.Pain<-"Yes"}
    if(as.numeric(svm.train$DM[i])==0){  DM<-"No"}else{DM<-"Yes"}
    if(as.numeric(svm.train$HTN[i])==0){ HTN<-"No"} else{HTN<-"Yes" }
    if(as.numeric(svm.train$Atypical[i])==0){ Atypical<-"No" }else{
        Atypical<-"Yes"}
    data<-data.frame("Typical.Chest.Pain"=Typical.Chest.Pain,"Age"=svm.
        train$Age[i],"Atypical"=Atypical,"FBS"=svm.train$FBS[i],"HTN"=
        HTN,"DM"=DM,"EF.TTE"=svm.train$EF.TTE[i], "K"=svm.train$K[i],"BP
        "=svm.train$BP[i],"ESR"=svm.train$ESR[i], "TG"=svm.train$TG[i],"
        Region.RWMA"=svm.train$Region.RWMA[i])
    app$setInputs("Typical.Chest.Pain"=Typical.Chest.Pain,"Age"=svm.
        train$Age[i], "Atypical"=Atypical,"FBS"=svm.train$FBS[i],"HTN"=
        HTN, "DM"=DM,"EF.TTE"=svm.train$EF.TTE[i], "K"=svm.train$K[i],"
        BP"=svm.train$BP[i],"ESR"=svm.train$ESR[i], "TG"=svm.train$TG[i
        ],"Region.RWMA"=svm.train$Region.RWMA[i])
    app$setInputs(button="click")
    output <- app$getValue(name="pred")
    a<-paste0("",output)
    numextract <- function(string){ str_extract(string, "\\-*\\d+\\.*\\
        d*")}
    num<-as.numeric(numextract(a))
    if(num>=0.5){result<-"Cad"  }else{result<-"Normal"}
    Typical.Chest.Pain<-as.factor(as.numeric(ifelse(Typical.Chest.Pain
        =="Yes",1,0)))
    Atypical<-as.factor( ifelse(Atypical=="Yes","Y","N"))
```

```
    HTN<-as.factor(as.numeric(ifelse(HTN=="Yes",1,0)))
    DM<-as.factor(as.numeric(ifelse(DM=="Yes",1,0)))
    data<-data.frame("Typical.Chest.Pain"=Typical.Chest.Pain,"Age"=svm.
        train$Age[i], "Atypical"=Atypical,"FBS"=svm.train$FBS[i],"HTN"=
        HTN, "DM"=DM,"EF.TTE"=svm.train$EF.TTE[i], "K"=svm.train$K[i],BP
        =svm.train$BP[i],ESR=svm.train$ESR[i], "TG"=svm.train$TG[i],"
        Region.RWMA"=svm.train$Region.RWMA[i])
    pr=round(predict(svm.1,data[,!names(data) %in% c("Cath")],type = "
        prob")$Cad,2)
    if(pr>=0.5){test<-"Cad"}else{test<-"Normal"}
    expect_equal(result,test)
    i=i+1})
app$stop()
```

a. Expected: OK:10 Failed :0

b. Result: As expected.

```
             ==> Testing R file using 'testthat'

  OK F W S | Context
  10       | Test Shiny app [7.6 s]

== Results =====================================
Duration: 10.1 s

OK:       10
Failed:   0
Warnings: 1
Skipped:  0

Test complete
```

3.4.4 Test: Website Integration

We use the `testthat` and `shinytest` packages for integration testing to test whether the shiny app works well. We first run the code `recordTest(here("website/FIT3164team1cad prediction"))` to create a tests folder in the same folder as the App file. Then the integration testing code in the `TestApp.R` file tests our shiny app.

Integration testing code: **TestApp.R**

```
library(testthat)
library(shinytest)
library(here)
test_that("Application works", {
  expect_pass(testApp(here("website/FIT3164team1cadprediction")))
})
```

a. Expected: OK :1 Failed :0

b. Result: As expected.

```
            ==> Testing R file using 'testthat'

  OK F W S | Context
   1       | Test_App [5.0 s]

== Results =====================================
Duration: 5.4 s

OK:       1
Failed:   0
Warnings: 0
Skipped:  0

Test complete
```

3.4.5 Test: Website Input Sanitation

We defined the acceptable type for most input variables to be numbers, so the user can only enter numbers in the input file. If they try to enter any categories other than numbers, the input box should not display the value they entered. We create a select list of input control for the rest variables, so the user can only choose items from this list. The user can also choose not to enter or select a value.

Manual Test:

a. What: All possible inputs do not throw an error.

b. How: Try many different inputs manually.

c. Expected: Inputting non-numeric values are blocked. A null entry also does not cause an error.

d. Result: As expected. The website's UI will not let non-numeric characters be entered, and a null value calls for the website to use a predetermined default value.

3.4.6 Test: Website Execution Paths

Manual Test: Our website has two main execution paths - one for manual input and one for file input.

a. What: All execution paths are run.

b. How: Changing the output of one path temporarily, then attempting both types of input.

c. Expected: Inputting a file gives one message while manually typing results gives another message.

d. Result: As expected. Although, the website needs to be refreshed to give a new manual input after a file has been uploaded.

3.5 Useability

We tested our website for useability through third parties, providing the link to users who are unfamiliar with the project with little explanation. We aimed to have the website be usable even without consulting the user guide.

The initial website had acronyms for input titles and 1/0 as options. We received feedback that this was confusing and made the website hard to understand. Following this, we changed all input names to descriptions with a suggested range, inputs to 'Y'/'N' and added a short description.

We then conducted a second round of testing, with participants using a range of devices. The website worked on both mobile and PC, with no distortion or resizing issues. However, a bug was encountered where a null value in a numerical field caused the website to throw an error. One user remarked that "I don't see why it has to be Y as opposed to Yes". From these surveys, we again clarified the input options, having 'Yes' and 'No' as options instead of 'Y' and 'N'. We also fixed the bug identified.

Finally, we received feedback from Daniel during the code demonstration. He stated that it could be clearer when the website has filled in missing values with default data, and that we should validate the file input to check for multiple patients.

3.6 Limitations discovered

3.6.1 Model

Due to all the manual code (wrangling and training), our training cannot be easily retrained with a different data set. We decided this was an acceptable outcome when we wrote the code, as it would save a lot of time. Furthermore, new data sets are rare in the research space, with the previous one being many years old, and so we would not have to change data often, if at all.

We also only trained a small collection of classifiers and ensemble methods. Perhaps we have missed a stronger model for this context which would have resulted in test cases with better than 90% test accuracy. Our models all plateaued at that threshold, which is likely a limitation in our ability and knowledge within the domain. See Limitations in the Code Report for more details.

Although cross validation was used during training, the testing set has remained the same throughout all tests. This may lead to some models performing better than others by chance, if the structure of the test data is particularly suited for one model over another.

3.6.2 External Connection

The website can still be a bit unclear, regarding when a user has omitted data. We did not have time to follow up on some comments received during testing, but it is not a major issue in terms of useability.

The website also needs to be refreshed if the user decides to use manual input after uploading a file. Again, this is a minor issue and would rarely occur, and so has been left in the current version of the software.

3.7 Recommendations

Future adaptations of this project should explore custom models tailor made for this domain, as our test cases indicate that stock models have difficulty surpassing 90% test accuracy.

In future tests, the testing data set could be sampled multiple times instead of used in its entirety, to reduce inter-model variance due to the structure limitation above.

The API should also be publicly hosted in future updates. Finally updates to clarity in the website should be made, resolving the issues raised in the Limitations section above, based off the feedback obtained in the useability tests.

3.8 Discussions of testing process

We acknowledge that all testing is flawed, and it is impossible to test for everything. We have made every effort attempts to test everything, but we may have missed bugs due to their obscurity. Further testing would be a public beta test, allowing many users to try out the website and model and report any bugs found through everyday use.

3.8.1 Omitted tests

We have also deliberately omitted some tests, for example input sanitation for the API, as it was not implemented into the code. This was because the data is not saved anywhere and so it is not vulnerable to attacks, and hence inputs can simply be handled by the front end.

We also did not test our models with new data sets from different regions of the world or produced from different experiments. This was done in the interest of time, although could be conducted in future to compare robustness.

Finally, we did not verify the individual models in the `caret` package, as we trusted that the model worked the way the documentation said it would. This was a reasonable assumption as `caret` is a very popular package and would been tested by many other users.

3.9 Conclusion

The tests have indicated that our project was a success, on the whole. We met the key deliverables and functionality set out in the project proposal, which was to train a collection of models, create ensembles, and present the best performing model as an API and website.

However, one criteria that we failed to meet was to make an ensemble model stronger than our individual models. This was partly due to our inexperience in ensembling models, as well as the relative strength of our individual models. Our literature review found that experts in the field created models with 94% accuracy, so it was improbable that our models would surpass that. Hence, with some models attaining 90% accuracy, there was little chance of an ensemble performing better.

Despite that, our team is happy with the deliverables produced, and have learnt a lot throughout the project.