FIT3164: Software Project
Code Report & Test Report

Cassandra Elliott (28786777), Julia Paterson () & Katie Polus (28792785)

Code Report

Code Documentation:

Please note that the documented source code is in a file called Documentation.R.

Development Considerations:

Some important considerations for the testing and performance evaluations stem from the dataset, model, and server. Performing calculations on the go can cause the application to lag too much. As a result, the team decided to hardcode some of the values in order to minimise the number of continuous calculations. For example, the mean for each point is not calculated within the server, rather default values were determined prior and input by the developers. Furthermore, the bagging model is an ensemble method, meaning that it is more computationally intensive than a smaller or simpler model. This also has the potential to lag the server, as a result, all important calculations and functions are performed in the back end rather than server side.

Limitations & Potential Improvements:

The final web application was designed with the purpose of being an informative and interactive app. Although this was achieved, there still exist limitations and improvements that the team would aim to develop assuming a greater timeline.

The main limitation of the app is its scalability. The lack of open-source data with the appropriate attributes and complete values was very limited and after much consideration, the team decided to use a single data source for the development of the predictive model.

The data used contains 303 instances which is not enough to result in a model which can be used alongside formal medical diagnosis. Furthermore the model does not include lifestyle and behavioural factors such as diet or exercise that are also significant predictors of heart disease.

In order to improve the scalability of the app, the team would train and test the model on a significantly larger amount of data. Moreover, the team would specifically look for sources which contain a wider demographic of patients.

Much of the feedback, both from team members, tutors and test participants involved mentions of stylistic choice on the UI. For example, the size of the text in most places should be increased for a better user experience. Additionally, the team would like to improve the output of the predictive model. As this is the main component of the application, it should be more engaging - one of the suggestions made by the test participants was to potentially have a pop-up displaying the result as opposed to the current - render text underneath the button. The team would like to take this suggestion on board in future development.

Finally, the team is subscribed to the free version of Shiny.io server. Although unable to implement the web application through the online server, it remains functional within Rstudio. If able to publish it online, it would be quite limited as there is a maximum of 25 hours per week of online access. In order to make the app more accessible to users, provided the funding, the team would aim to scale the project by investing in an upgraded account.

End User Guide

Introduction:

The purpose of this project is to develop a web-based application that is informative and interactive. It aims to educate users on the prevalence of heart disease and common features or attributes which can impact one's likelihood of being diagnosed with cardiovascular disease. The application allows the user to test their own likelihood against multiple factors and receive a result with an accuracy of 81.3%. The application also aims to inform the user of how the model was developed and why it was chosen.

The development team does not have any formal medical training and are not authorized to provide actual medical advice. All results from this application should be taken as informative and do not precede formal medical diagnosis.

1. Accessing the App

Assuming that the app is already running, the user can skip to step 2. Alternatively, the user can access the app through the following URL:
https://www.shinyapps.io/fit3164datascienceprojectteam3
(Please note, that although the team deployed the app, it is currently not accessible via the URL - discussion on this point has been made in the conclusion of this report.)

2. Navigating through the App

The user should navigate through the tabs in the order they appear to engage to the fullest extent with the experience. The user may choose to try the interactive components on each page, viewing the definitions of each variable, plotting different variables against each other to see which are more/less highly correlated, and reading about the different models and selection process.

The app is designed such that those without a medical background are able to take part with both the content of the application as well as the interactive components, including the predictive model.

3. Trying the Predictive Model

Navigating to the tab 'Predictive Model', the user shall select between the two options: 'Basic' and 'Advanced'. These two options relate to the type of data required for the model to be run effectively.

For users with minimal medical information, it is suggested that they select the 'Basic' model as no recent blood test information or more complex medical records are required. It should be known that for fields which are left unchanged by the user, the average value from the data for these variables will be used by default.

For users who have more medical information, it is encouraged that they try the 'Advanced' model. Once again, if values are unknown, the default will be the average of the used dataset.

Technical User Guide:

1. Downloading the Project

The completed project is stored on GitHub and can be downloaded by the user via their terminal. Instructions are shown below:

1. **Install Git**
   If the user already has access to 'git' via their terminal, skip to step 2. Otherwise, the user should install git by typing the following command directly into their terminal:
   ```
   sudo dnf install git-all
   ```
   To check that it is installed type the following command:
   ```
   git version
   ```
2. **Download repository**
   Now that git is installed locally, the user should be able to clone the project repository via the following command:
   ```
   git clone https://github.com/poluskg/FIT3164.git
   ```

2. Software Requirements

The application is to be run in the RStudio IDE. It is recommended that the user has a minimum of R version 3.6.3 in order to run the app to the best capacity. Instructions for upgrading or downloading the latest version of RStudio can be found through the following link: https://rstudio.com/products/rstudio/download/

3. Package Installation

The application requires certain packages for rendering particular visualisations, styles and layouts. If the user has not previously installed these packages, it is recommended that they do so prior to running the app for the first time. RStudio will prompt you on how to install the required packages via your console.

4. Opening the Project

Once downloaded, the user will find the zipped project in their 'Downloads' folder. Opening RStudio, the user should navigate to the 'File' button and select 'Open Project'. The user should go to their 'Downloads' folder (or whichever directory they cloned the repository into), open the folder and select 'FIT3164.Rproj' to open the project in their IDE.

5. Running the App

Now that the project is opened and all software requirements and packages are installed, the user should open the file 'ui_helper.R' and navigate to line 54. Now, the user needs to load in the dataset by running this single line using the Run button on the top right:

```
data <- read.csv("Z-Alizadeh_sani_dataset.csv", header = TRUE)
```

Once the data has been loaded, the user should click on either one of 'ui.R' or 'server.R' files then select 'Run App' in the top right hand corner of their IDE. For a more structured app experience, the user can now refer to the 'End User Guide' for details on navigation.

Testing Report

As we have been using a flexible agile methodology, testing has occurred throughout the duration of our project. After each feature was created, it was tested immediately, and all features were tested again after any updates. The main components of the project are the classification model and the web application, which have required different testing processes.

Model Testing

We have used multiple different methods to test the accuracy of the models we have implemented. These tests were mainly manual, as we used R to make calculations and predictions for each model separately. The data was randomly split into 70% training data and 30% testing data.

````
Separate testing and training data
```{r}
load("z.Rdata")

set.seed(1111) #random seed

# 70% of the data is training, 30% is testing
train.row = sample(1:nrow(z), 0.7*nrow(z))
data.train = z[train.row,]
data.test = z[-train.row,]

```
````

After training the models on the training data, predictions were made using the test data, and confusion matrices were constructed. Accuracy, sensitivity and specificity were calculated using the equations below.

	Cad	Normal
Cad	True Positive (TP)	False Positive (FP)
Normal	False Negative (FN)	True Negative (TN)

$$ACCURACY = \frac{TN + TP}{TN + TP + FN + FP}$$
$$SENSITIVITY = \frac{TP}{TP + FN}$$
$$SPECIFICITY = \frac{TN}{TN + FP}$$

We further assessed the goodness of fit of each model by calculating the area under the receiver operating characteristic curve (ROC) for each model using the pROC package in R.

```
#Confusion matrix to check accuracy
table(nb_Predictions,data.test$Cath)

# Calculate area under the ROC curve
roc_obj <- roc(as.numeric(data.test$Cath), as.numeric(nb_Predictions))
auc(roc_obj)
```

```
nb_Predictions Cad Normal
        Cad     24      1
        Normal  36     30
Setting levels: control = 1, case = 2
Setting direction: controls < cases
Area under the curve: 0.6839
```

The table below shows the results of the model testing.

Model	**Accuracy**	**AUC ROC**	**Sensitivity**	**Specificity**
Naive Bayes	59%	0.6839	0.4	0.968
Decision Tree	75.82%	0.7387	0.828	0.636
Random Forest	79.12%	0.7169	0.781	0.833
XgBoost	79.1%	0.7481	0.815	0.731
SVM	80.2%	0.7487	0.809	0.783
Bagging	81.31%	0.796	0.864	0.719

Unit Testing

Our project did not require many functions, and the functions we have made are quite simple. Below are the unit tests for the functions that we coded. They use the testthat R package.

CalculateBMI

CalculateBMI is a simple function that calculates the users body mass index from their weight and height. Using the 'testthat' package in R, we constructed the unit test below. We have tested the accuracy of the function only. We do not need to see how the function handles negative values or non numeric inputs as the user interface only accepts numeric values for weight and height that are within a set range. All tests passed successfully.

```
Unit testing for calculateBMI
This will test the accuracy of the calculateBMI function, by comparing the outputs of the function to
expected BMI
inputs: calculateBMI() function,
        these weight/height combinations: (118, 167), (124.6, 217), (49.88472,153.7757)
expected output: No output if all the tests are correct
```

```{r}
test_that("BMI of user",
          { expect_equal(calculateBMI(118, 167), 42.310588404)
            expect_equal(calculateBMI(124.6, 217), 26.4605321837)
            expect_equal(calculateBMI(49.88472,153.7757),21.0956112462)

          })
```

isObese

isObese calculates if a user is obese using BMI. BMI is calculated using calculateBMI(), and therefore all of the possible BMIs will be a real number, so we do not need to test for handling invalid inputs. All tests passed.

```
Unit testing for isObese
This will test the accuracy of the isObese function, by comparing the outputs of the function to expected B
inputs: isObese() function,
        these BMIs: 42, 30, 29.5, 26
expected output: No output if all the tests are correct
```

```{r}
test_that("BMI of user",
          { expect_equal(isObese(42), "Y")
            expect_equal(isObese(30), "Y")
            expect_equal(isObese(29.5),"N")
            expect_equal(isObese(26),"N")

          })
```

getGender

This function changes the user input for gender to the same format the model takes. The input is given from a radio button with the inputs "Female" or "Male", so we do not need to test the handling of invalid inputs. All tests passed.

```
Unit testing for getGender
This will test the accuracy of the getGender function, by comparing the outputs of the function to expected gender
inputs: getGender() function,
        "Female"
        "Male"
expected output: No output if all the tests are correct
```

```{r}
test_that("Gender of user",
          { expect_equal(getGender("Female"), "Fmale")
            expect_equal(getGender("Male"), "Male")

          })
```

changeValues

This function changes the user input of “Yes” or “No” and changes it to 0/1 format, so the model can interpret it. The input is given from a radio button with the inputs “Yes” or “No”, so we do not need to test the handling of invalid inputs. All tests passed.

```
Unit testing for changeValues
This will test the accuracy of the changeValues function, by comparing the outputs of the function to expected
output
inputs: changeValues() function,
        "Yes"
        "No"
expected output: No output if all the tests are correct
```

```{r}
test_that("Changing values",
          { expect_equal(changeValues("Yes"), 1)
            expect_equal(changeValues("No"), 0)

          })
```

getBBB

This function translates the user input from the radio button for BBB which is Left, Right or None, to the format the model takes (LBBB, RBBB, N). Again as it the user input is only from the radio button we did not test invalid inputs. All tests passed.

```
Unit testing for getBBB
This will test the accuracy of the getBBB function, by comparing the outputs of the function to expected output
inputs: getBBB() function,
        "Left"
        "Right"
        "None"
expected output: No output if all the tests are correct
```

```{r}
test_that("BBB",
          { expect_equal(getBBB("Left"), "LBBB")
            expect_equal(getBBB("None"), "N")
            expect_equal(getBBB("Right"), "RBBB")

          })
```

getVHD

This function translates the user input from the radio button for VHD which can take the values of Mild, Moderate, None or Severe to the format the model takes (mild, Moderate, N, Severe). Again as it the user input is only from the radio button we did not test invalid inputs. All tests passed.

```
Unit testing for getVHD
This will test the accuracy of the getVHD function, by comparing the outputs of the function to expected output
inputs: getVHD() function mild if Mild, Moderate if Moderate, N if None, Severe if Severe
        "Mild"
        "Moderate"
        "None"
        "Severe"
expected output: No output if all the tests are correct
```

```{r}
test_that("VHD",
          { expect_equal(getVHD("Mild"), "mild")
            expect_equal(getVHD("None"), "N")
            expect_equal(getVHD("Moderate"), "Moderate")
            expect_equal(getVHD("Severe"), "Severe")

          })
```

Shiny Interactive Scatterplot

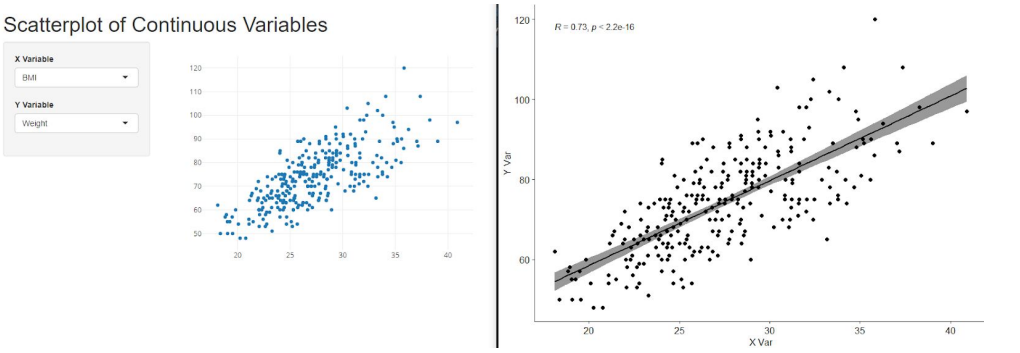
This plot was tested manually by visually comparing it to a pearson correlation plot generated with ggplot2, which includes a correlation coefficient and p value that can be used for additional verification. As the inputs would only be from the user selection of a particular continuous variable, we did not test invalid inputs. We also check that the file for the shiny ui exists for robustness. A selection of, but not all, test graphs are included below.

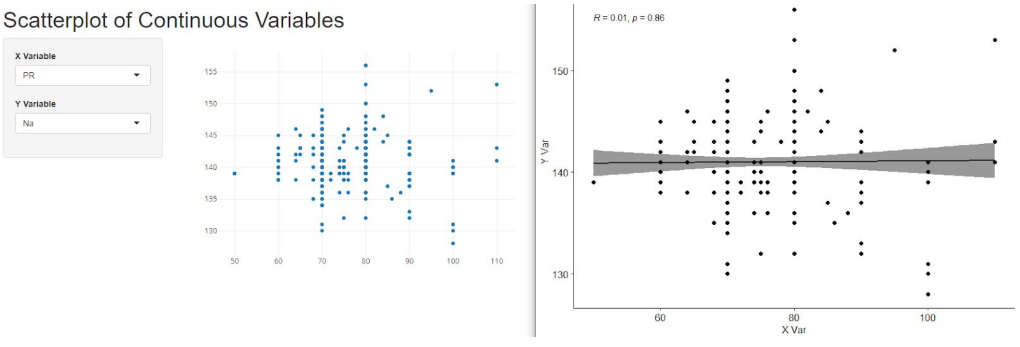
All tests passed.

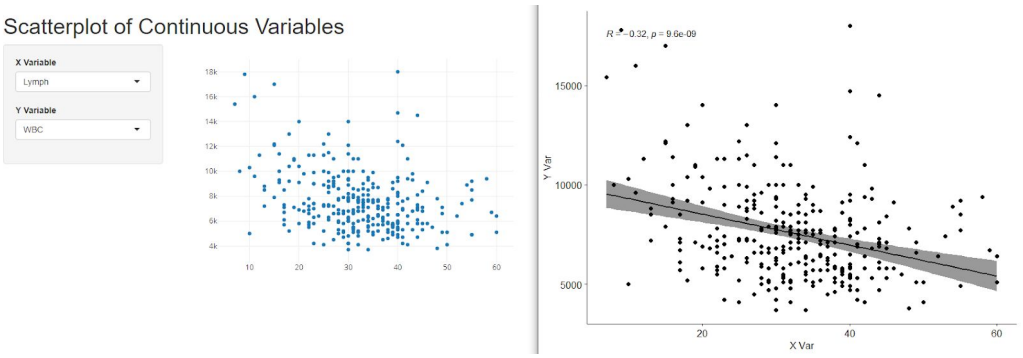
```
library(ggplot2)
library(ggpubr)
library(testthat)

#check shiny file where scatterplot exists for robustness
file.exists('dataOverview_ui.R')

#manually create scatterplots for variable combinations to confirm visually
attach(za)
ggscatter(za, x = "Lymph", y = "WBC", add = 'reg.line',
          conf.int = TRUE, cor.coef = TRUE, cor.methods = 'pearson',
          xlab = 'X Var', ylab = 'Y Var')
```







Integration Testing

The overall functionality of the web application was tested manually, as our program is very user oriented. The tests were conducted in the following sections:

Navigating Between Tabs

The navigation between the five tabs was tested by manually clicking on each tab, from each of the other tabs. This test was successful as the program acted as expected; each tab loaded correctly and there was no interference from other parts of the program.

Introduction and Acknowledgement Pages

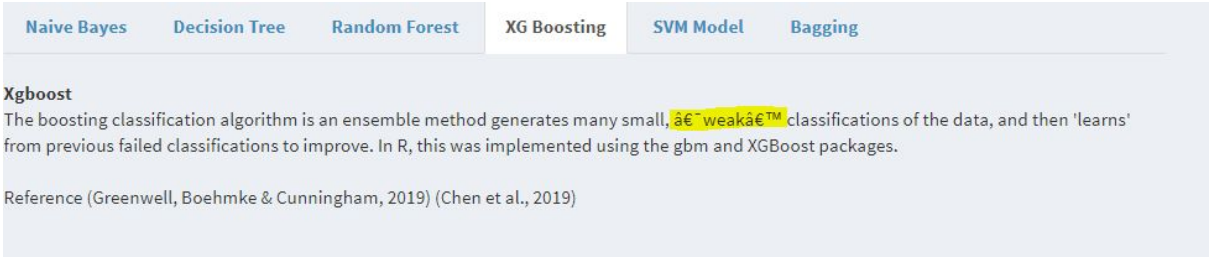
These pages were manually verified to function properly by a visual inspection.

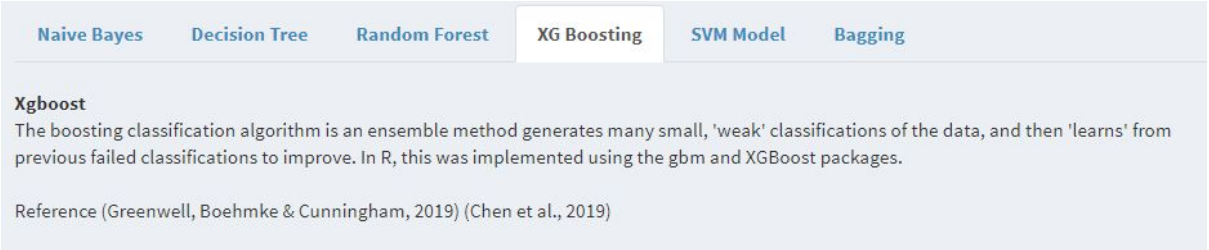
Data Overview

Each of the variables in the “select a variable” page were visually inspected to ensure the output was correct. Additionally, each variable pair in the interactive scatterplot was tested to ensure a meaningful graph was produced. All tests passed.

Modelling Process

We inspected each model tab to ensure that the model definitions were working correctly. We found that there was an error within the XG boost definition caused by a different quotation mark style. After changing the quotation mark style in modellingProcess.R, this issue was resolved.





Predictive Model

In previous tests of the predictive model, we had found that the web application crashed after pressing the "GET RESULTS" button. After debugging, we found that this was due to a few of the user inputs being saved as one variable name in the UI, but in Server.R they were being called using a different name. This resulted in the prediction not working and the application crashing.

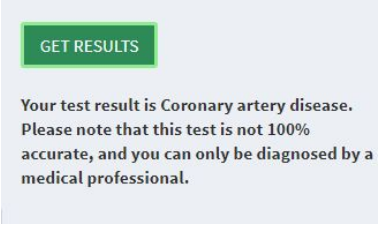
For the final test, we are inputting certain values taken from the data set that we know should give a CAD or normal result. By doing this, we manually verified that the predictive model gives accurate predictions when integrated into the web application, as our model testing has already proved that our model is 81.31% accurate.

Basic Model

The following values taken from row 11 of the data set are input to the basic predictive model section:

Age: 58, Weight: 75, Height: 168, Sex: Male, Smoking Status: Ex smoker, Typical Chest Pain: Yes, Hypertension: Yes, Blood Pressure: 170, Pulse Rate: 70.

The expected output is that he does have coronary artery disease.



This test was successful. The following tests on the basic model are summarised in this table.

Row number in data	**Inputs to UI**	**Expected Result**	**Actual Result**
17	Age: 41, Weight: 60, Height: 169, Sex: Male, Smoking Status: Smoker, Typical Chest Pain: No, Hypertension: No, Blood Pressure: 130, Pulse Rate: 80	Normal	CAD
114	Age: 51, Weight: 80, Height: 170, Sex: Female, Smoking Status: Non smoker, Typical Chest Pain: No, Hypertension: Yes, Blood Pressure: 120, Pulse Rate: 70	Normal	Normal
44	Age: 60, Weight: 80, Height: 165, Sex: Female, Smoking Status: smoker, Typical Chest Pain: No, Hypertension: Yes, Blood Pressure: 140, Pulse Rate: 70	CAD	CAD

For the basic model 3/4 tests were successful. Due to our model being only 81.31% accurate, we can be sure that the model integration is working properly.

Advanced Model

The advanced model was tested by the same process as the basic model. The results are summarized in this table.

Row number in data	Inputs to UI	Inputs continued	Expected Result	Actual Result
78	Age "70" Weight "60" Length "144" Sex "Fmale" BMI "28.93519" DM "0" HTN "1" Current.Smoker "0" EX.Smoker "0" FH "0" Obesity "Y" CRF "N" CVA "N" Airway.disease "N" Thyroid.Disease "N" CHF "N" DLP "N" BP "190" PR "90" Edema "0" Weak.Peripheral.Pulse "N" Lung.rales "N" Systolic.Murmur "N" Diastolic.Murmur "N" Typical.Chest.Pain "0" Dyspnea "N" Function.Class "0"	LVH "N" Poor.R.Progression "N" BBB "N" FBS "130" CR "0.7" TG "210" LDL "168" HDL "43" BUN "12" Atypical "Y" Nonanginal "N" Exertional.CP "N" LowTH.Ang "N" Q.Wave "0" St.Elevation "0" St.Depression "0" Tinversion "0" ESR "12" HB "12.7" K "5.3" Na "142" WBC "5400" Lymph "40" Neut "55" PLT "170" EF.TTE "45" Region.RWMA "0" VHD "mild"	CAD	CAD
301	Age "48" Weight "77" Length "160" Sex "Fmale" BMI "30.07812" DM "0" HTN "1" Current.Smoker "0" EX.Smoker "0" FH "1" Obesity "Y" CRF "N" CVA "N" Airway.disease "N" Thyroid.Disease "N" CHF "N" DLP "N" BP "130" PR "70"	Atypical "N" Nonanginal "Y" Exertional.CP "N" LowTH.Ang "N" Q.Wave "0" St.Elevation "0" St.Depression "0" Tinversion "0" LVH "N" Poor.R.Progression "N" BBB "RBBB" FBS "83" CR "1" TG "93" LDL "112" HDL "42" BUN "13" ESR "20" HB "12.8"	Normal	Normal

	Edema "0" Weak.Peripheral.Pulse "N" Lung.rales "N" Systolic.Murmur "N" Diastolic.Murmur "N" Typical.Chest.Pain "0" Dyspnea "N" Function.Class "0"	K "4" Na "140" WBC "9000" Lymph "35" Neut "55" PLT "279" EF.TTE "55" Region.RWMA "0" VHD "N"		
44	Age "75" Weight "66" Length "156" Sex "Male" BMI "27.12032" DM "0" HTN "1" Current.Smoker "0" EX.Smoker "0" FH "0" Obesity "Y" CRF "N" CVA "N" Airway.disease "Y" Thyroid.Disease "N" CHF "N" DLP "N" BP "130" PR "70" Edema "1" Weak.Peripheral.Pulse "N" Lung.rales "Y" Systolic.Murmur "Y" Diastolic.Murmur "N" Typical.Chest.Pain "0" Dyspnea "Y" Function.Class "0"	Atypical "Y" Nonanginal "N" Exertional.CP "N" LowTH.Ang "N" Q.Wave "0" St.Elevation "0" St.Depression "0" Tinversion "0" LVH "N" Poor.R.Progression "N" BBB "N" FBS "90" CR "1.3" TG "190" LDL "180" HDL "29" BUN "21" ESR "16" HB "11.8" K "3.1" Na "138" WBC "9200" Lymph "18" Neut "75" PLT "214" EF.TTE "50" Region.RWMA "0" VHD "Severe"	CAD	CAD

For the advanced model 3/3 tests were successful. As we have already tested our bagging model against the test data set, we are satisfied that the integration of the advanced model was successful.

Usability Testing

Due to stay at home orders being in place for the majority of this project, we have only been able to test the usability of our project on people within our households. We tested the version of the interface that was seen in the week 10 code demonstration. The final web application takes into account user feedback. Below are the results of these tests.

User 1:

User Testing
Date: 23/10
User: Karen

Introduction
This is a web application that is intended to inform people about coronary artery disease. There are different sections of the web application, including a predictive model that can estimate if the user is likely to have heart disease or not. We will instruct you to do a few tasks on this application, then we have some questions we would like you to answer.

Tasks
(The application will have already been launched and the user will start from the 'Introduction' section.)

1. Read through the information on the 'Introduction' section of the application.
2. Navigate to the 'data overview' section
3. Select the blood pressure variable, and read the definition of blood pressure
4. Now open the 'Predictive Model' section
5. Change the age to 55
6. Change the gender to male
7. Under current smoker select yes
8. Press get results
9. Read the outcome of this prediction
10. Now change the model type from 'basic' to 'advanced'
11. Change any of the variables that you would like to change, then press get results
12. Read the outcome of this prediction
13. Navigate to the acknowledgements section

Questions	User Answer
How did you find navigating between pages?	Generally fine, getting back to the main page was harder from the advanced model.
What did you think about the information on the introduction section (the first section you started on)?	Very interesting.
How was using the data overview section?	Font too small on all pages, otherwise interesting.
What did you think of the predictive model?	User friendly.
What, if anything, do you think that this application is missing?	Page up/page down button, back button.

Do you have any other comments?	It could be more attractive to look at.

Overall, Karen found the application easy to use, but her main concern was that the text was too small in some parts, and some parts of the navigation difficult. We will remedy this by increasing the font size in the sections that she noted and putting more instructions about navigation on the top of each section. Some of her criticisms, such as adding a back button and return button were valid, but we will not have the time or ability to implement them in the R shiny format. We will also include more images to make the UI more attractive to look at, including a graph of the ROC area under the curve for each model.

User 2:

User Testing
Date: 23/10
User: Andrew

Introduction
This is a web application that is intended to inform people about coronary artery disease. There are different sections of the web application, including a predictive model that can estimate if the user is likely to have heart disease or not. We will instruct you to do a few tasks on this application, then we have some questions we would like you to answer.

Tasks
(The application will have already been launched and the user will start from the 'Introduction' section.)

14. Read through the information on the 'Introduction' section of the application.
15. Navigate to the 'data overview' section
16. Select the blood pressure variable, and read the definition of blood pressure
17. Now open the 'Predictive Model' section
18. Change the age to 55
19. Change the gender to male
20. Under current smoker select yes
21. Press get results
22. Read the outcome of this prediction
23. Now change the model type from 'basic' to 'advanced'
24. Change any of the variables that you would like to change, then press get results
25. Read the outcome of this prediction
26. Navigate to the acknowledgements section

Questions	User Answer
How did you find navigating between pages?	Navigation was simple and easy to follow. I had no problems with the overall usability.
What did you think about the information on the introduction section (the first section you started on)?	It was interesting and considering that I don't know anything about heart disease, it was useful to read prior to testing the model. It gave me context which I liked.
How was using the data overview section?	It was fine, the definitions were a good feature, it wasn't overly informative and I don't think that it detracted or added heaps to the overall application.

What did you think of the predictive model?	It was cool! A little bit anticlimactic though - maybe you could have done a pop-up for the result or something a little more engaging/visible.
What, if anything, do you think that this application is missing?	I don't know if it is missing anything to be honest...I like that it educates the user as they progress through the app. Maybe you could edit the design to look a little nicer.
Do you have any other comments?	I thought it was a really interesting project, nothing more to add. Well done.

Andrew is younger than Karen so possibly is more open to smaller text and navigating a web application. He has a background in design and business so his criticism on certain UI components was particularly helpful and the team will endeavour to make suitable changes.

User 3:

User Testing
Date: 24/10
User: Tim

Introduction
This is a web application that is intended to inform people about coronary artery disease. There are different sections of the web application, including a predictive model that can estimate if the user is likely to have heart disease or not. We will instruct you to do a few tasks on this application, then we have some questions we would like you to answer.

Tasks
(The application will have already been launched and the user will start from the 'Introduction' section.)

27. Read through the information on the 'Introduction' section of the application.
28. Navigate to the 'data overview' section
29. Select the blood pressure variable, and read the definition of blood pressure
30. Now open the 'Predictive Model' section
31. Change the age to 55
32. Change the gender to male
33. Under current smoker select yes
34. Press get results
35. Read the outcome of this prediction
36. Now change the model type from 'basic' to 'advanced'
37. Change any of the variables that you would like to change, then press get results
38. Read the outcome of this prediction
39. Navigate to the acknowledgements section

Questions	User Answer
How did you find navigating between pages?	It was fine.
What did you think about the information on the introduction section (the first section you started on)?	Scary.

How was using the data overview section?	It made sense.
What did you think of the predictive model?	So this is what you learn in computer science! Good job. It makes sense.
What, if anything, do you think that this application is missing?	Better graphics.
Do you have any other comments?	No.

Tim has a background in games programming so is somewhat familiar in developing UIs and interactive tools. He was easily able to navigate and interact with the application however wasn't particularly engaged with it. As our team does not have any design experience we cannot do much to improve the current aesthetics but have taken the general feedback into account. Tim's review was very beneficial in confirming that someone without a health or statistical background could understand the app and models.

Limitations of the Software

The testing process revealed many limitations of our software. From the user testing, we received unanimous feedback that the design and visuals of our application could be improved. However, because none of us have studied UI/UX before, we were unable to improve this aspect.

Recommendations for Improvement

As mentioned in the code report section, we could improve our application by making the visuals more appealing and modern. We would also like to train our model on a larger data set, so that we can create more accurate predictions, and potentially validate our results with medical and/or statistical experts. We could also conduct usability tests on a larger sample of both laypeople and medical professionals to determine if the tone of our app is appropriate for both. Furthermore, obtaining data about individual participant's lifestyle factors such as diet could help us develop a more nuanced and personalized prediction.

Limitations of Testing Process

A large portion of our testing process was manual testing, which may not be as thorough as programmed tests. We had limited testing resources due to the continuation of the Melbourne lockdown and so only a rather limited number of end users have tested the app and provided feedback. However, this was enough to identify some common themes for potential improvements, predominantly relating to graphics and UI. It is also difficult to test if our application is accessible for all users. Given more time and resources, or if in person meetings were possible, these limitations could be addressed however unfortunately this has not been the case.

Conclusion

Ultimately, the team was successful in fulfilling their original project goal: To develop an interactive and informative app on the topic of heart disease, which contains a fully integrated predictive model developed to a high accuracy. We have ensured this model is accessible to lay-people in order to promote a novel exploratory approach to the understanding of heart disease and its risk factors.

We were successfully able to perform a variety of unit, integration, model and usability testing to identify any bugs and ensure accuracy of our models and web app.

Although the app works locally and there were no issues whilst testing, unfortunately, the team were unable to successfully deploy the application via the Shiny.io server. As far as limitations and improvements to the project, a priority for the team would be to deploy the application such that anyone can access it online.

Despite the obvious difficulties faced during the project's life cycle - working remotely due to COVID19 restrictions, facing occasional technical issues (both hardware and software) - the team were able to overcome this and meet the project deadline.

References:

Alfaro, E., Gamez, M., & Garcia, N. (2013). adabag: An R Package for Classification with Boosting and Bagging. Journal of Statistical Software, 54(2), 1-35. Retrieved from http://www.jstatsoft.org/v54/i02/

Breiman, L., Cutler, A., Liaw, A., and Wiener, M. (2018). randomForest: Breiman and Cutler's Random Forests for Classification and Regression. R package version 4.6-14. https://CRAN.R-project.org/package=randomForest

Chen, T., He, T., Benesty, M., Khotilovich, V., Tang, Y., Cho, H., Chen, K., Mitchell, R., Cano, I., Zhou, T., Li, M., Xie, J., Lin, M., Geng, Y., and Li, Y. (2019). xgboost: Extreme Gradient Boosting. R package version 0.90.0.2. https://CRAN.R-project.org/package=xgboost

CSS Text. (n.d.). Https://Www.W3schools.Com/Css/Css_text.Asp. https://www.w3schools.com/css/css_text.asp

Greenwell, B., Boehmke, B., Cunningham, J., & GBM Developers. (2019). gbm: Generalized Boosted Regression Models. R package version 2.1.5. https://CRAN.Rproject.org/package=gbm

Meyer, D., Dimitriadou, E., Hornik, K., Weingessel, A., & Leisch, F. (2019). e1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien. R package version 1.7-3. https://CRAN.R-project.org/package=e1071

Parashchenko, R. (2018, January 30). *Software testing test report templates.* StrongQA. https://strongqa.com/qa-portal/testing-docs-templates/test-report

Shiny - Tutorial. (n.d.). Https://Shiny.Rstudio.Com/Tutorial/. https://shiny.rstudio.com/tutorial/

Shiny - observeEvent. (n.d.). Https://Shiny.Rstudio.Com/Reference/Shiny/1.0.5/ObserveEvent.Html.

Therneau, T., Atkinson, B., & Ripley, B. (2019). rpart: Recursive Partitioning and Regression Trees. R package version 4.1-15. https://CRAN.R-project.org/package=rpart

Unit Testing in R Programming. (2020, July 23). GeeksforGeeks. https://www.geeksforgeeks.org/unit-testing-in-r-programming/